TREBALL DE FI DE GRAU

Programació d'una aplicació mòbil - PILLBOX

Autor: Pau Casas Garcia

Directora: Ana Gabriela Zuñiga Zarate

Titulació: Grau en Multimèdia

Convocatòria: Curs 2014-2015

1. Índex dels continguts

2. Introducció

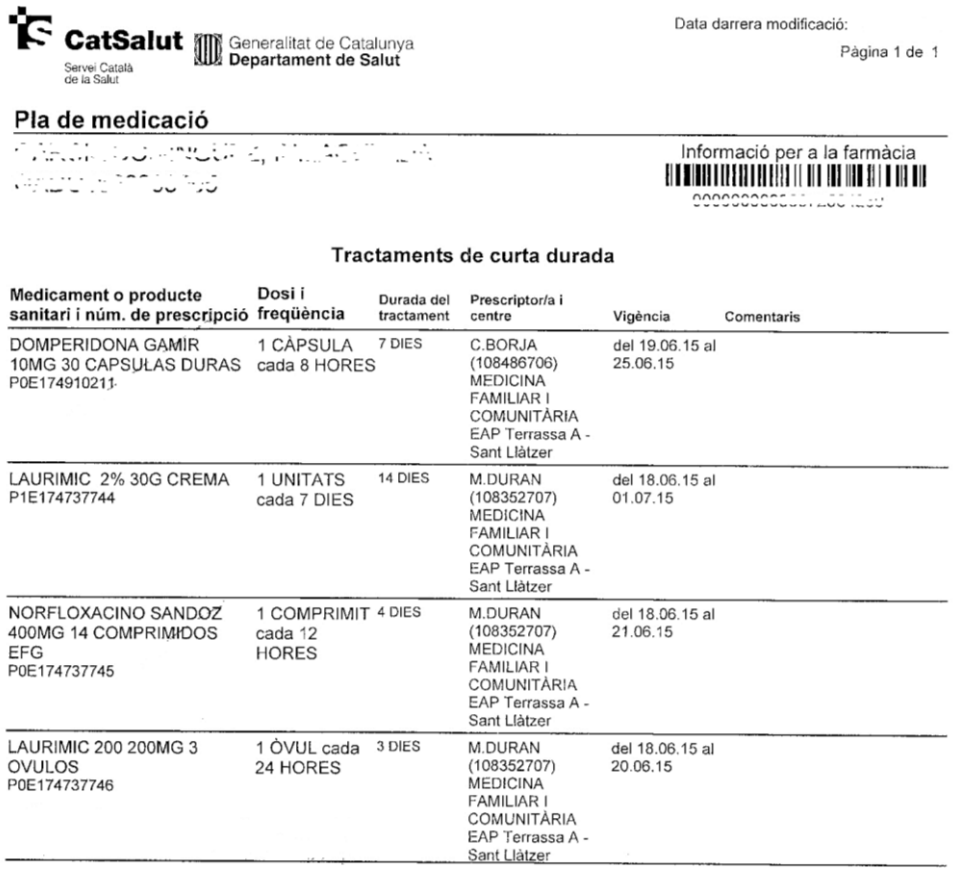
La idea de Pillbox va néixer fruit d'una anècdota. La meva àvia necessitava ajuda per organitzar-se tots els medicaments que havia de prendre. La persona encarregada de fer-ho era la meva mare, que li col·locava tots els medicaments en un pastiller dividit en Matí/Tarda/Vespre. A cada divisió hi posava la dosi exacta de càpsules de cada medicament i després la meva àvia només havia de prendre les pastilles que hi haguessin en funció de l'hora del dia. Aquest sistema però, no és del tot infal·lible. Es poden confondre fàcilment dos medicaments amb colors de pastilla similars i col·locar una dosi equivocada a la caixa. Tampoc hi havia garanties per falta de medicaments o control del que es prenia o no, tot i que el sistema era útil per les dos. Un dia, la caixeta va caure i totes les pastilles es van barrejar i va ser un caos per la meva àvia perquè no hi havia manera d'identificar-les totes i saber on anaven. Va ser llavors quan vaig pensar en tota la gent que ha d'organitzar un pla de medicació d'aquesta manera perquè necessita prendre molts medicaments. A partir d'aquí vaig relacionar el tema amb els telèfons mòbils intel·ligents i les seves aplicacions.

Vaig pensar que un pla de medicació dinàmic en Android podria complir una mateixa funció i fins i tot millorar-la, perquè la mateixa aplicació podria avisar a la persona de qualsevol notificació, com ara un medicament acabat o l'hora que ha de prendre la següent dosi.

D'aquesta manera vaig voler emprendre un projecte relacionat amb la programació d'aquesta aplicació, intentant afegir totes les funcionalitats que els meus coneixements d'objectes en programació m'ho permetessin i, en cas contrari, investigar solucions per fer-ho una realitat.

2.1 Objectius

L'objectiu principal del projecte és programar una aplicació mòbil des de zero amb la funció de pla de medicació i gestor de pastilles a la vegada. Fem una ullada de com és un pla de medicació actual:



CatSalut Servei Català de la Salut — Generalitat de Catalunya, Departament de Salut

Data darrera modificació:

Pàgina 1 de 1

Pla de medicació

Informació per a la farmàcia

Tractaments de curta durada

Medicament o producte sanitari i núm. de prescripció	Dosi i freqüència	Durada del tractament	Prescriptor/a i centre	Vigència	Comentaris
DOMPERIDONA GAMIR 10MG 30 CAPSULAS DURAS P0E174910211	1 CÀPSULA cada 8 HORES	7 DIES	C.BORJA (108486706) MEDICINA FAMILIAR I COMUNITÀRIA EAP Terrassa A - Sant Llàtzer	del 19.06.15 al 25.06.15	
LAURIMIC 2% 30G CREMA P1E174737744	1 UNITATS cada 7 DIES	14 DIES	M.DURAN (108352707) MEDICINA FAMILIAR I COMUNITÀRIA EAP Terrassa A - Sant Llàtzer	del 18.06.15 al 01.07.15	
NORFLOXACINO SANDOZ 400MG 14 COMPRIMIDOS EFG P0E174737745	1 COMPRIMIT cada 12 HORES	4 DIES	M.DURAN (108352707) MEDICINA FAMILIAR I COMUNITÀRIA EAP Terrassa A - Sant Llàtzer	del 18.06.15 al 21.06.15	
LAURIMIC 200 200MG 3 OVULOS P0E174737746	1 ÒVUL cada 24 HORES	3 DIES	M.DURAN (108352707) MEDICINA FAMILIAR I COMUNITÀRIA EAP Terrassa A - Sant Llàtzer	del 18.06.15 al 20.06.15	

Vull elaborar una aplicació que permeti introduir dades de medicaments a una llista com la de la figura i poder gestionar-les després, per tant, fer-la dinàmica. Si ens hi fixem, és un pla molt detallat d'un pacient en concret. En el meu cas, opino que no hauria de ser tant estricte al ser una aplicació personal. Els camps del centre i prescripció per exemple, són camps innecessaris en un gestor de medicaments. Per tant, la meva idea és agafar els camps essencials a l'hora de prendre càpsules d'un pla de medicació i convertir-lo en un gestor de medicaments Android.

2.2 Calendari

(Només es tracta d'una captura, mostraré tot el calendari als annexos)

	Nombre de tarea	Duración	Comienzo	Fin
1	**Treball de Final de Grau**	**244 días**	**lun 06/10/14**	**jue 10/09/15**
2	**Memòria**	**244 días**	**lun 06/10/14**	**jue 10/09/15**
3	**1. Introducció**	**244 días**	**lun 06/10/14**	**jue 10/09/15**
4	1.1 Objectius	1 día	lun 06/10/14	lun 06/10/14
5	1.2 Calendari	243 días	mar 07/10/14	jue 10/09/15
6	**2. Filosofia de la tecnologia**	**3 días**	**mar 10/02/15**	**jue 12/02/15**
7	2.1 Introducció	1 día	mar 10/02/15	mar 10/02/15
8	2.2 Tecnologia vinculada sanitat	2 días	mié 11/02/15	jue 12/02/15
9	**3. Estudi de mercat**	**11 días**	**mar 07/10/14**	**mar 21/10/14**
10	3.1 Introducció	1 día	mar 07/10/14	mar 07/10/14
11	3.2 Objectiu investigació	1 día	mié 08/10/14	mié 08/10/14
12	3.3 Anàlisi situació actual	2 días	vie 10/10/14	lun 13/10/14
13	3.4 Anàlisi DAFO	5 días	mar 14/10/14	lun 20/10/14
14	3.5 Públic objectiu	1 día	mar 21/10/14	mar 21/10/14
15	3.6 Referents	1 día	mar 21/10/14	mar 21/10/14
16	**4. Producció**	**118 días**	**lun 23/03/15**	**mié 02/09/15**
17	4.1 IDE Android Studio	3 días	lun 23/03/15	mié 25/03/15
18	4.2 Accions prèvies	2 días	jue 26/03/15	vie 27/03/15
19	4.3 Creació d'una activitat	3 días	lun 30/03/15	mié 01/04/15
20	4.4 Maquetació	7 días	lun 15/06/15	mar 23/06/15
21	**4.5 Programació**	**13 días**	**lun 17/08/15**	**mié 02/09/15**
22	4.5.1 Activitats	10 días	lun 17/08/15	vie 28/08/15

El calendari, juntament amb la bibliografia, és un dels apartats que he realitzat al llarg de tot el treball. Ha patit moltes variacions, per problemes d'organització i la pròrroga que vaig demanar per fer possible l'entrega després d'estar treballant i cursar primer del grau superior en informàtica a la vegada mentre feia el TFG. Tot i així, l'estructura és clara i directe. He dividit el treball en dos grups: Memòria i Aplicació. Aplicació engloba des de la primera idea que vaig tenir sobre com seria fins la seva programació i desenvolupament. La memòria descriu tots els passos seguits i ensenya tot el procés de redactat des de la introducció fins als annexos. Tal i com podem observar, la memòria s'ha realitzat en diferents parts i espais de temps. En un principi, vaig redactar els objectius i l'estudi de mercat, seguit per la Filosofia de la Tecnologia i la part de Producció que no englobava la meva aplicació en sí. La resta, es va redactar un cop acabada l'aplicació i la seva programació.

3. Filosofia de la tecnologia

3.1 Introducció

La tecnologia és una aplicació sistemàtica del coneixement científic o altres tipus de coneixements lligats a treballs pràctics. Quan parlem de tecnologia no estem parlant només de maquinària. Les màquines son els resultants més visibles de la tecnologia i per això sovint es relacionen com a conjunt indispensable. Però res mes lluny de la realitat: la tecnologia és la col·lecció de tècniques, mètodes o processos utilitzats en la producció de productes o serveis i fins i tot en el compliment d'objectius, com ara recerques d'investigació científica. Per tant, la tecnologia és coneixement. Aquest coneixement es pot incrustar a les màquines, ordinadors, dispositius i fàbriques per realitzar tasques pràctiques amb menor temps i esforç.

La qualitat de vida de les persones millora al poder realitzar tasques de forma més ràpida i eficaç amb menys esforç, ja sigui pels mateixos individus o amb eines com les que he enumerat abans les quals tenen el coneixement de la tecnologia "incrustat".

Hi ha molts sectors diversos on la tecnologia i el seu coneixement pot afectar, ja sigui directe o indirectament. Alguns camps es centren en millorar la qualitat de vida de les persones, altres en salvar o allargar vides i fins i tot altres per facilitar la "destrucció" d'individus, com la tecnologia armamentista. Tots i cada un dels sectors de la tecnologia es basen en la teoria científica. És a dir, els treballs d'investigació permeten millores tecnològiques. Cada avenç en tecnologia es manté per les pròximes generacions i permet un creixement exponencial per nous coneixements i tecnologies. Les nostres vides han estat radicalment transformades per la innovació digital en pocs anys. Si observem la vida que portem nosaltres i la comparem amb la dels nostres pares/avis veurem que aquesta afirmació és certa. El creixement imparable de la tecnologia pot fer el mateix amb les següents generacions amb canvis encara més notables.

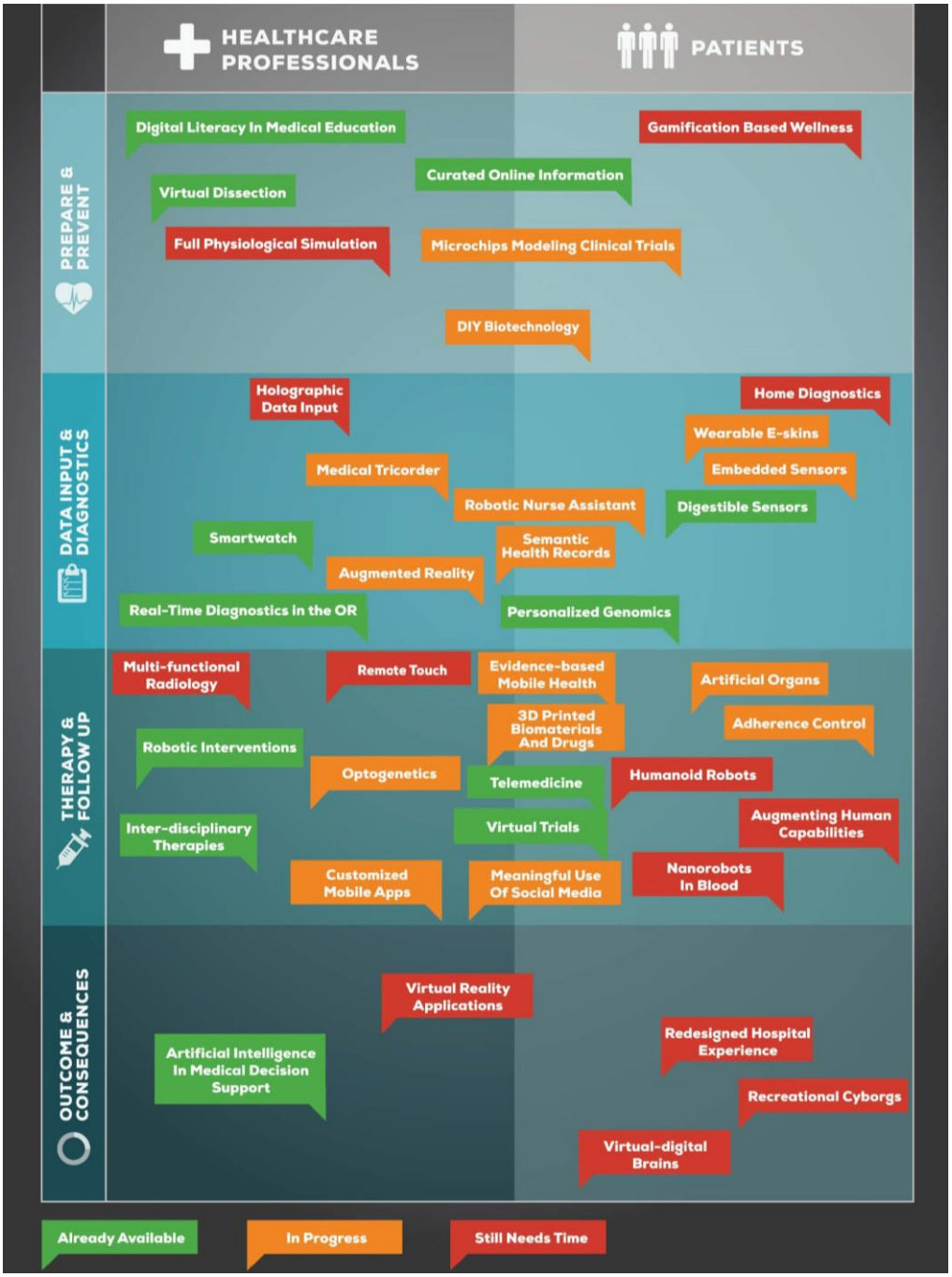
Si tenim en compte que els objectius de la tecnologia poden variar en funció dels sectors els quals volem aplicar-la, es podria dir que l'objectiu final de la tecnologia és millorar la qualitat de vida de les persones.

Actualment, cada cop més dispositius i màquines apliquen coneixements tecnològics que ens fan la vida més fàcil. Els mòbils intel·ligents, les tabletes, els ordinadors portàtils... cada vegada són més potents i accessibles a tothom. No és d'estranyar que les aplicacions per mòbil hagin generat un mercat nou que pateix un creixement també exponencial.

Més enllà del tema econòmic, em centraré en investigar l'àmbit sanitari de la tecnologia i com aquests coneixements tecnològics permeten una qualitat de vida superior si els ajuntem amb les aplicacions mòbils.

3.2 El sector de la tecnologia vinculada als serveis sanitaris

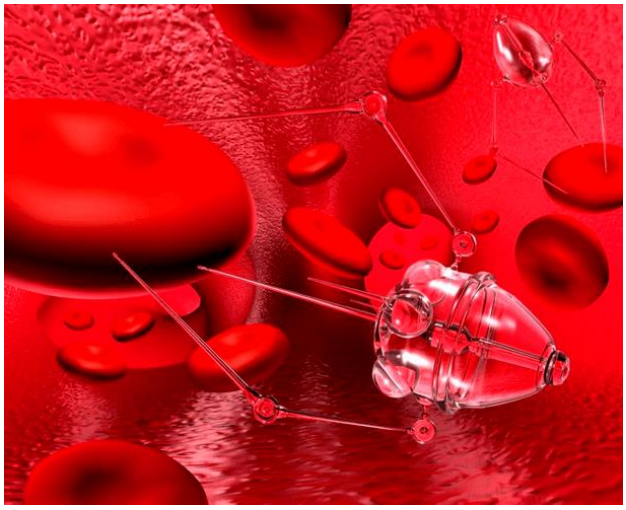
Les innovacions tecnològiques digitals han permès millorar la sanitat en general. Des de un aparell de radiologia fins a un GPS portàtil per les ambulàncies, els avanços també es fan notar en el camp de la medicina i permeten actuar més ràpidament i salvar vides. També podem monitoritzar millor qualsevol malaltia i els accessos a metges e informació mèdica és més fàcil gràcies a internet.



Actualment es treballa en tecnologies mèdiques com les que es mostren al gràfic que encara que necessitin més temps, ajudaran als metges i pacients de tot el món. Un exemple de tecnologia en desenvolupament és els nano robots a la sang. Aquests robots microscòpics tindrien sensors propis i sistemes de propulsió i permetrien portar oxigen, reparar cèl·lules i destruir bactèries, apart de subministrar tractaments com la quimioteràpia mil vegades més efectiva que l'actual mitjançant drogues i no tindrien els efectes secundaris que tenen avui dia.

Tots aquests avenços e investigacions requereixen inversions molt elevades i es fan a gran escala per la medicina en general. Ara em centraré en la tecnologia recent i accessible a tothom que permet funcionalitats molt útils amb un cost relativament baix, com són les aplicacions mòbils.

La revolució recent dels telèfons mòbils intel·ligents ha obert un nou món de possibilitats en tots els sectors. En l'àmbit de la salut, ja existeixen aplicacions que permeten al usuari utilitzar dispositius portàtils per accedir a la seva informació mèdica, monitoritzar les seves constants vitals, diagnosticar-se des de casa seva i portar un ventall ampli de tasques mèdiques a la butxaca. És tracta d'una nova era de medicina, la de poder fer tot això a temps real i fins i tot adquirir imatges d'ultrasò del cor i abdomen d'un mateix o un infant d'una embarassada.



La imatge de l'esquerra mostra nano robots a la sang. La imatge de la dreta són sensors comestibles en forma de pastilla que mesuren i envien dades sobre la salut del individu i les mostra a través d'una aplicació per mòbil.

4. Estudi de mercat

4.1 Introducció

Les aplicacions mòbils creixen com a indústria i ja formen part del dia a dia de moltes persones. L'App Store va néixer el juliol de 2008 amb 500 aplicacions disponibles. Als pocs anys la botiga d'Apple arribava ja al milió d'apps i Play Store ja les sobrepassava per a dispositius Android. Observem doncs que les descàrregues d'aplicacions mòbils en ambdues plataformes creixen a un ritme vertiginós que és directament proporcional a les vendes de "smartphones" i tauletes, i totes dues van superar els 50 bilions de descàrregues al llarg de 2013. iOS i Android (de les companyies Apple i Google, respectivament) són els principals sistemes operatius actualment dels dispositius mòbils. Per tant, la majoria d'aplicacions son creades pels dos sistemes operatius per arribar a la majoria de clients existents en tot el món. Es podria dir doncs, que monopolitzen les ventes d'aplicacions mòbils a través de botigues online d'aplicacions com Play Store (Android) i App Store (iOS). Android pren avantatge en quota de mercat però Apple lidera els ingressos procedents de les botigues d'aplicacions; son els diners que paguen els usuaris per utilitzar les apps, ja sigui per mitjà de pagar per descàrrega (model "Premium"), per subscripció o bé pagar per opcions avançades un cop l'usuari s'hagi descarregat gratuïtament l'aplicació (model "Freemium"). Aquestes son les principals estratègies de monetització existents en el món de les aplicacions mòbils, però encara es busca la millor manera de monetitzar el negoci.

És un mercat creixent doncs amb molta competència. Les empreses que s'encarreguen de crear aplicacions les vénen a les botigues online especialitzades de les quals hem parlat en funció del sistema operatiu pel que s'hagin creat (normalment les grans empreses fan una versió per Android i una altre per iOS).

4.2 Objectiu investigació

En el meu cas, estudiaré les aplicacions mòbils relacionades amb la salut, concretament amb el control de medicaments. El meu objectiu és facilitar la feina d'administració tant de la gent com dels metges. Vull conèixer aquest tipus d'aplicacions per millorar-les o crear-ne una de innovadora, ja que és un camp no gaire explotat. L'objectiu principal de la meva investigació és l'aplicació de la "Seguridad Social". De fet, son tres aplicacions diferents disponibles de forma separada: "Seguridad Social", "Cita Previa" i "Informe de Situación".

-Seguridad Social (App):

Es poden consultar els directoris, oficines, novetats i enllaçar amb pàgines web. Cal registrar-se com usuari, cosa que et permetrà fer consultes específiques sobre la teva situació i reservar una cita.

-Cita Prèvia (App):

Es tracta d'una aplicació senzilla que permet concertar cites amb els CAISS (Centros de Atención e Información de la Seguridad Social) i cancel·lar-les en cas de canvi d'agenda.

-Informe de Situación (App):

Es pot consultar l'últim informe de la teva situació laboral dins de la Seguretat Social, es a dir, si estàs de baixa, per quina empresa treballes, a quina província desenvolupes la teva activitat en cas de ser autònom...

Totes tres aplicacions son gratuïtes i disponibles per IOS i Android, amb les seves respectives versions per Tablet.

4.3 Anàlisi previ de la situació actual

Tenint en compte que estem parlant d'una aplicació mòbil i no d'una empresa, els recursos que s'han d'utilitzar són diferents:

-Anàlisi intern: Aquí entra en joc el meu coneixement de creació d'aplicacions Android. Pel que fa al cost econòmic, només n'hi haurà en cas de publicar l'aplicació a "google play". El cost és més de hores de feina i càrrega de treball. Disposo de tot el material per desenvolupar l'aplicació, a més de connexió a internet i softwares per desenvolupadors (Eclipse ADT o Android Studio faré servir pel codi).

-Anàlisi extern: Podríem dir que en aquest aspecte la competència és la principal qüestió externa. Les aplicacions de la Seguretat Social son gratuïtes i serveixen per més d'un sistema operatiu. Tinc pensat desenvolupar l'aplicació només per Android i, en cas de publicar-la, seria gratuïta també.

4.4 Anàlisis D.A.F.O.

Debilitats (factors interns):

-No sóc expert en programació d'Android, gairebé seria la meva primera aplicació.

-No faré versió per IOS, de Apple. La gràcia es fer una aplicació funcional i elegant, les versions son secundàries.

-La col·laboració de metges o centres hospitalaris son vitals per un funcionament total de la aplicació, per tema receptes o simplement informació dels pacients, tot i que hi ha altres solucions.

Amenaces (factors externs):

-Les aplicacions mòbils amb la salut com a temàtica estant molt de moda. De tota manera, en un àmbit diferent.

Fortaleses (factors interns):

-La meva aplicació serà unificada i millorarà la usabilitat de les aplicacions de la Seguretat Social.

-Molt temps per dedicar-hi.

Oportunitats (factors interns):

-La indústria de les aplicacions mòbils creix desmesuradament i la relació tecnologia-salut agrada molt a la gent i als usuaris en general.

-Les aplicacions de la Seguretat Social no son gaire usables i requereixen 3 instal·lacions diferents; la poca competència que hi ha no és gaire bona.

4.5 Públic Objectiu

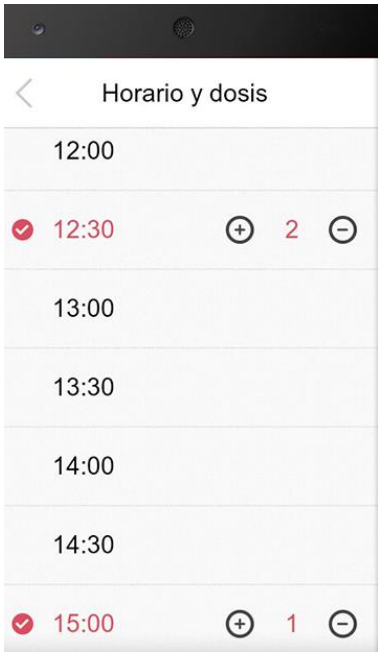
El meu gran públic objectiu són els usuaris d'Android. Específicament els usuaris amb malalties cròniques els quals necessiten un gestor de medicaments per facilitar la seva organització. Tothom que utilitzi un pastiller físic podrà aprofitar aquesta aplicació per tenir recordatoris constants i un control digital i portàtil a través del seu mòbil. La gent gran també pren molta medicació, normalment controlada per un pla de medicació que han de seguir. Una tercera persona sol ajudar amb aquest pla de medicació gestionant els medicaments de la gent gran, així que encara que aquest públic no sol ser usuari d'Android, aquesta tercera persona pot utilitzar l'aplicació tant per ella mateixa com per gestionar els medicaments d'aquestes persones grans.

4.6 Referents – Gestors de medicaments

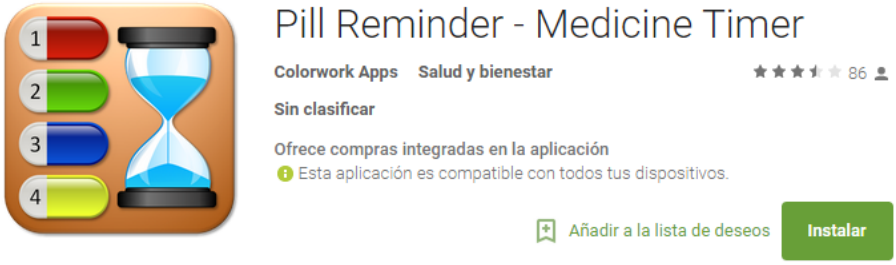
A mesura que ha anat passant el temps des de la meva proposta de Treball Final de Grau, han anat sortit noves aplicacions. La més important és "Mi Pill", les altres són poc conegudes i tenen poques descarregues. És potent i emergent al mercat actual i amb funcionalitats molt similars a la meva aplicació. És cert que el meu projecte ha anat evolucionant i variant al llarg dels mesos i la programació d'una aplicació amb una funcionalitat que cada cop s'explota més me'n alegra molt. La idea d'un gestor de medicaments no era nova, però si que ho era en el món de les aplicacions mòbils i estic content d'haver-ho escollit.

-Mi Pill – Meds Reminder

Es tracta d'un recordatori de medicaments simple i efectiu, el més conegut i descarregat de Play Store . El meu objectiu és programar totes les funcionalitats que facilitin una correcte gestió de medicaments, de forma més simple però eficaç i amb més opcions com ara un sistema de notificacions d'alerta o un temporitzador que indiqui quan pots tornar a prendre un medicament en funció del temps preestablert.

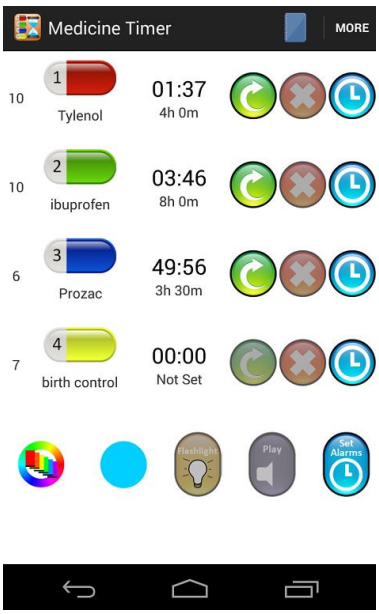


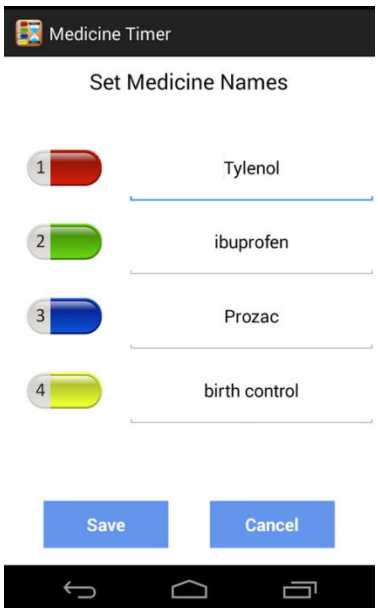
-Pill Reminder

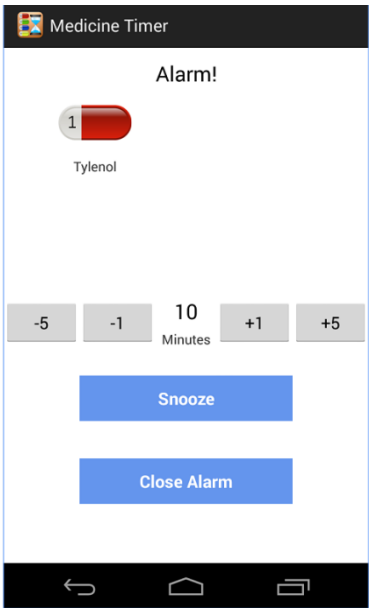


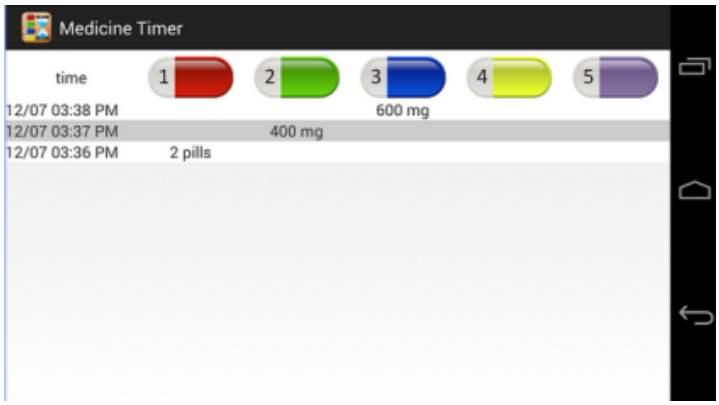
Aquest gestor de medicaments va aparèixer el febrer de 2014. És més bàsic que l'anterior i ofereix molt poca flexibilitat; Només permet introduir quatre medicaments a la vegada i té moltes opcions i botons que donen lloc a confusió. Tot i així, conté un sistema d'alarmes força bo i funciona correctament. Té una versió "landscape" tot i que millorable des del punt de vista de programació.

*Nota: a la versió actual permet afegir més medicament per un preu de 0,74€ cada un.

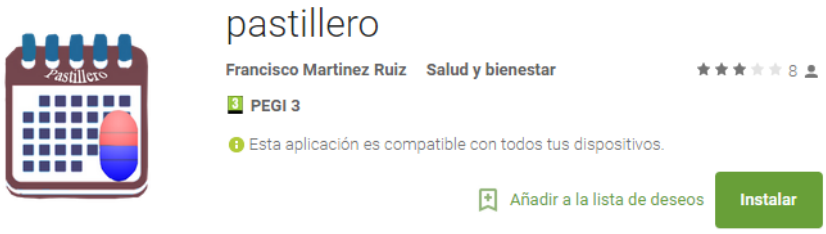




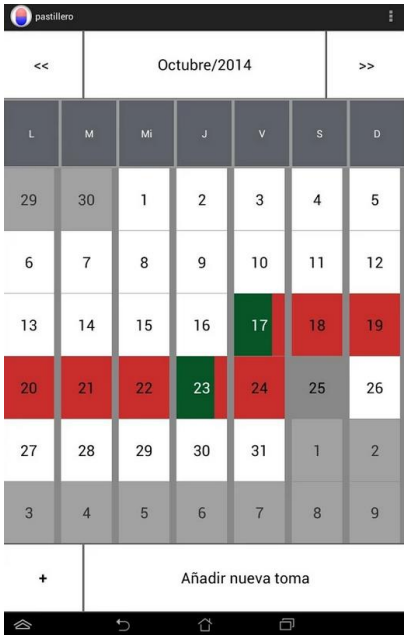


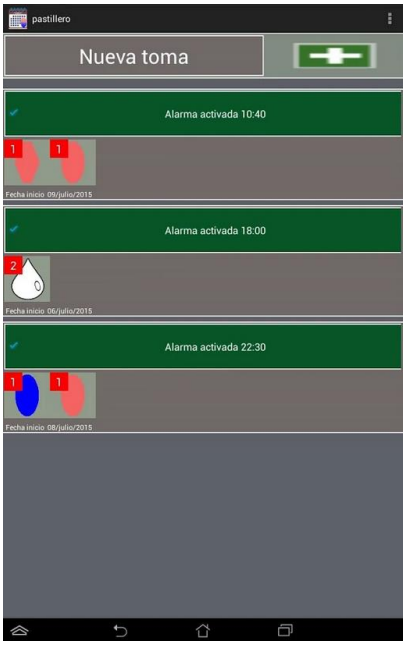


-Pastillero



Aquesta aplicació és molt recent: va ser publicada a Play Store el 05/09/2015. Guarda una imatge virtual de cada medicament que vols prendre i té un bon sistema d'alarma. Apart d'això, les pantalles es bloquegen i els elements de text es llegeixen malament en molts casos si el text que s'introdueix és mínimament llarg. Moltes vegades no agafa bé l'hora de prendre un medicament i no et deixa donar-lo d'alta per aquest problema.





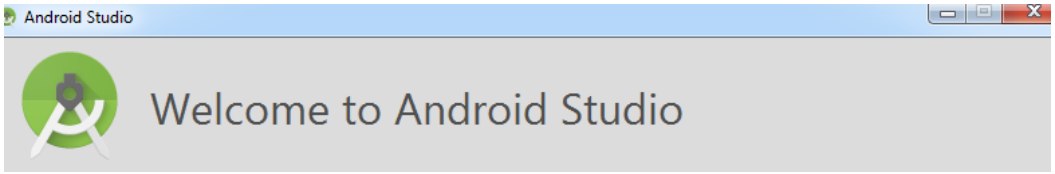
En definitiva, aquest tipus d'aplicacions encara poden millorar i aniran evolucionant amb el temps perquè encara hi ha poques opcions decents si volem tenir el nostre gestor de medicaments al mòbil. En termes de programació, m'agradaria un resultat similar a la millor aplicació al mercat del moment (Mi Pill – Meds Reminder).

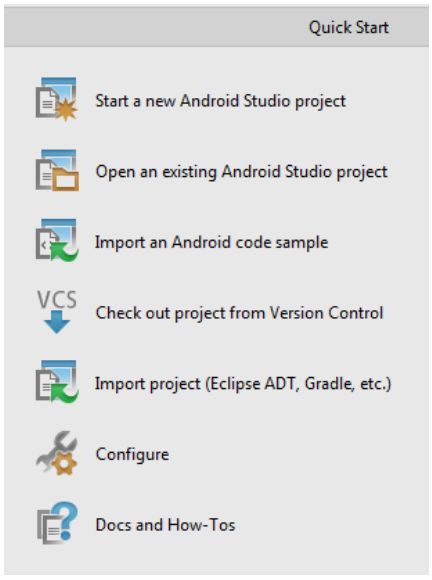
5. Producció

5.1 IDE utilitzat – Android Studio

Durant el meu quart curs de carrera vaig cursar l'assignatura optativa de Java i després la d'Android. En els dos casos, vàrem fer servir Eclipse com a interfície per programar en Java. Es tracta d'una eina potent i totalment gratuïta, però necessita un SDK i una extensió per poder programar aplicacions Android i està preparat per fer altres aplicacions (Applets, per exemple). Per això, vaig decidir utilitzar la beta del programa gratuït del gegant Google: Android Studio.

Android Studio és un programa totalment gratuït i està pensat exclusivament per fer aplicacions per mòbil. T'ofereix diferents plantilles d'activitats, com per exemple una aplicació amb lo bàsic per utilitzar google maps. És força intuïtiu i organitza millor el teu projecte. El programa va deixar de ser BETA als pocs mesos de començar el 2015.

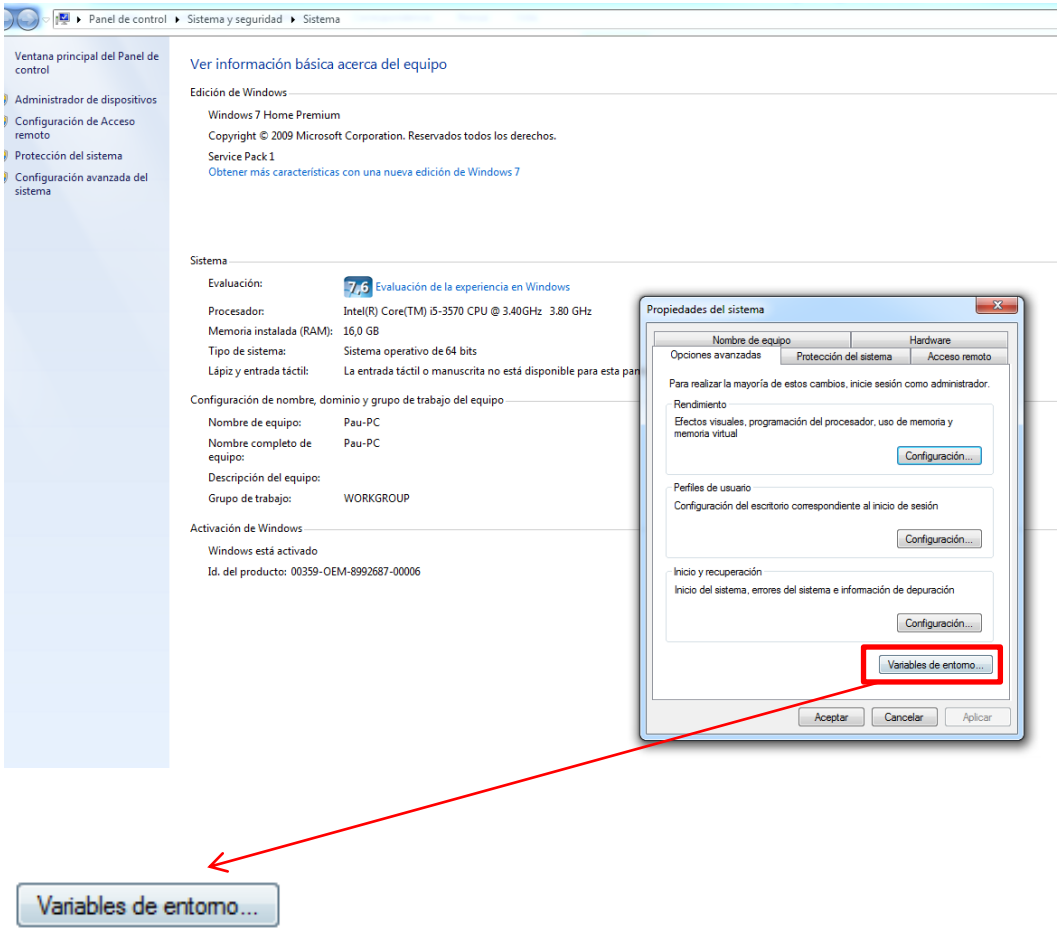




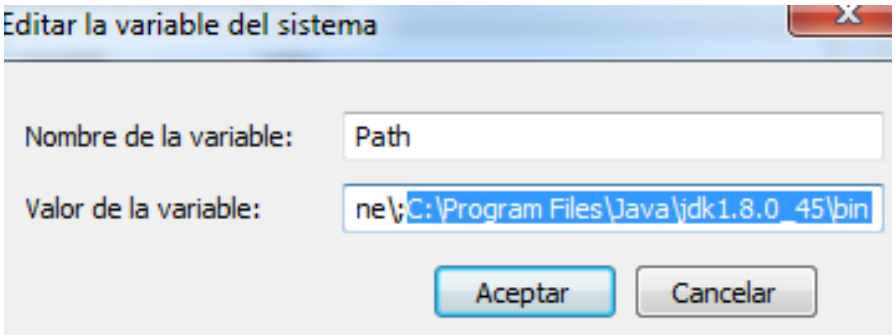
Durant el meu curs d'Android al Nicolau Copèrnic vaig estar programant amb aquest IDE i vaig aprendre a utilitzar-lo. Ara veurem exemples d'aplicacions que vaig fer allà per veure com és l'estructura d'un projecte en aquest programa i tot el que necessita per poder funcionar, com el JDK i l'SDK . Veurem també el codi Java (pel codi font) i el codi XML (pels layouts i l'estructura).

5.2 Accions prèvies

Per fer funcionar el programa, necessitem tenir Java (jdk) instal·lat al ordinador. Ens podem baixar el "jdk" de la pàgina oficial d'Oracle tal i com fèiem per fer funcionar també el programa Eclipse. Quan l'instal·lem, necessitem indicar la ruta al programa i guardar-la com una variable d'entorn del sistema:



Aquí haurem d'assignar la ruta on tinguem instal·lat Java a la variable "Path" del sistema:

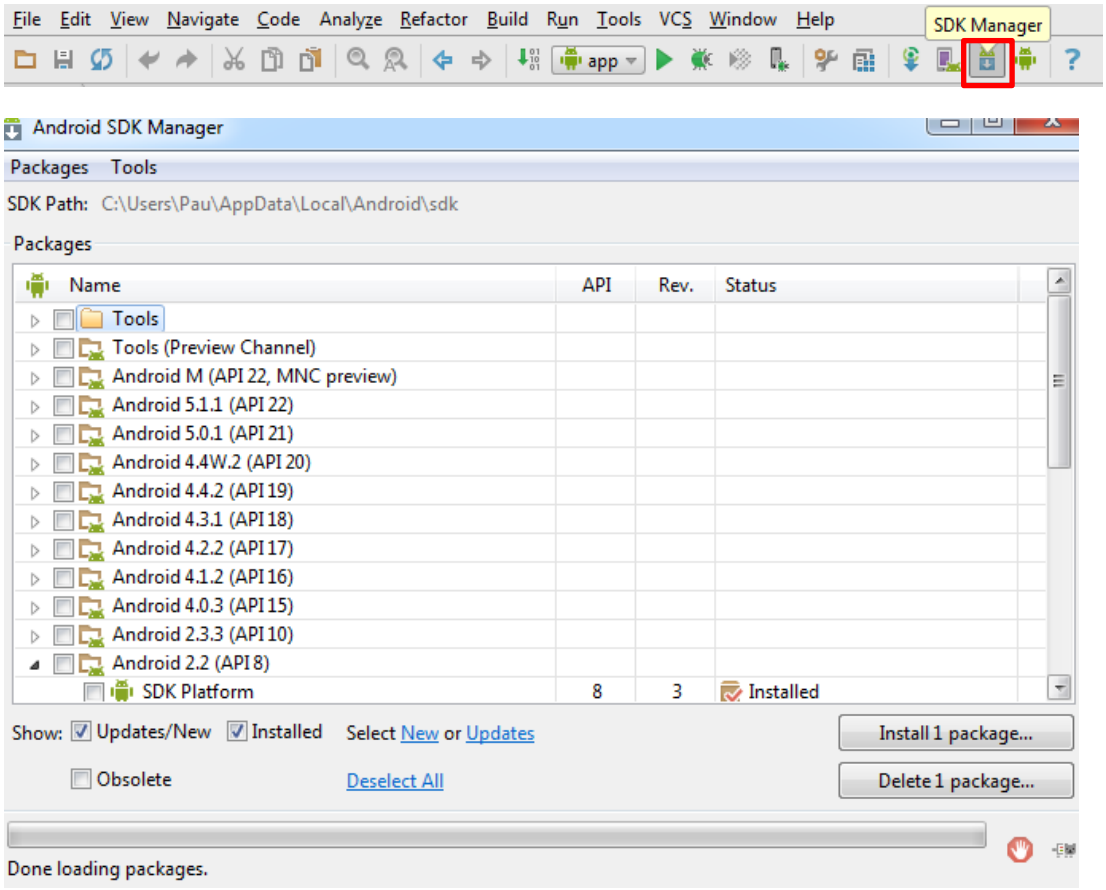


En el meu cas, la ruta és la que està marcada en blau.

D'aquesta manera ja podríem instal·lar correctament Android Studio però ens demanaria l'SDK d'Android, que són totes les llibreries que necessitem per crear aplicacions mòbils. Aquest pas també és necessari encara que utilitzéssim Eclipse.

Ens podem descarregar l'SDK de la pàgina oficial de google, la mateixa on ens hem descarregat l'Android Studio. Un cop instal·lat, el programa detecta automàticament la seva localització. En cas contrari, podem indicar-li la ruta on podria trobar-lo.

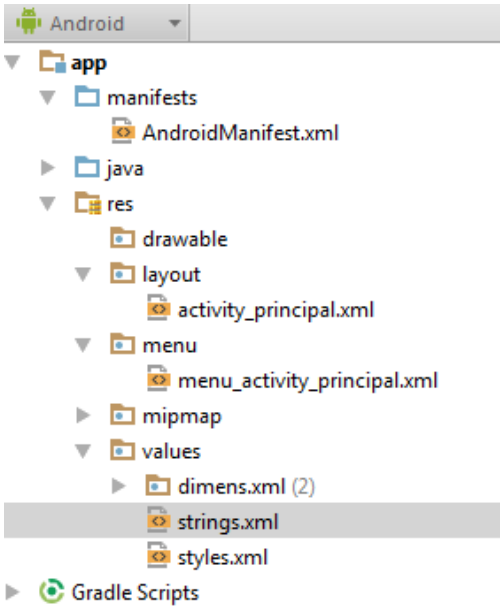
El programa disposa d'una opció que es diu "SDK Manager":



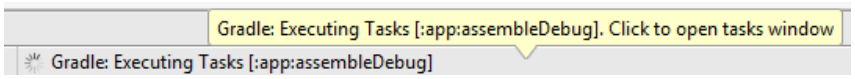
Aquesta aplicació et permet decidir quins paquets i eines vols instal·lar del SDK, així com la ruta on es troba en el teu ordinador. Jo he instal·lat totes les API's desde la 8 fins la 22, que passa per les versions d'Android 2.2 fins la 5.1.1 (Android Lollipop). Això em permetria fer aplicacions per qualsevol target de mòbil Android. De tota manera, una aplicació creada amb la API 15 significaria que totes les versions posteriors a ella la podran executar sense problema, com ja vàrem explicar, per això vaig decidir fer l'aplicació en aquesta versió. Recordem que la API 15 és la versió 4.0 d'Android tal i com es mostra a la imatge.

Ara ja tenim Android Studio ben configurat i ja podem començar a maquetar la nostre aplicació.

L'estructura d'un projecte està organitzat en carpetes:



"Gradle Scripts" son els arxius de codi generats automàticament per Android Studio que serveixen per poder executar la nostre aplicació. De fet, cada vegada que l'executem al nostre mòbil o a un dispositiu virtual com ara un emulador, recopila tota la informació del nostre codi i executa aquests scripts per fer tots els passos fins que veiem la sortida. L'execució no es fa possible quan aquests scripts fallen. Cada cop que executem la nostre aplicació per testejar-la ens mostra que el "debug" el genera amb els Gradle Scripts. Veiem-ho:



De tota manera, aquests scripts no haurem de modificar-los ni tocar-los. Només amb problemes de compatibilitat de versions vaig haver de canviar algunes línies de codi, però simplement els deixarem si funcionen correctament i no tindrem problemes.

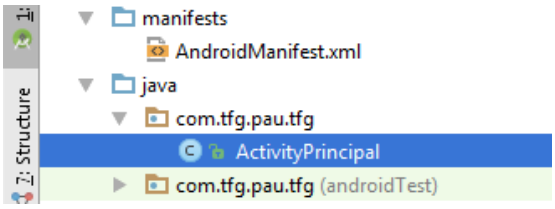
Del nostre projecte "app" veiem que ens divideix en 3 carpetes principals: Manifests, Java i Resources (res). L'Android Manifest és un document XML que conté tota la informació tècnica de la nostra aplicació. Sempre que vulguem crear una nova pantalla (Activity) o el logo/nom de l'aplicació, ho indicarem al manifest. Aquest arxiu per defecte té aquesta estructura:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tfg.pau.tfg" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="TFG Pau"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".ActivityPrincipal"
            android:label="TFG Pau" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

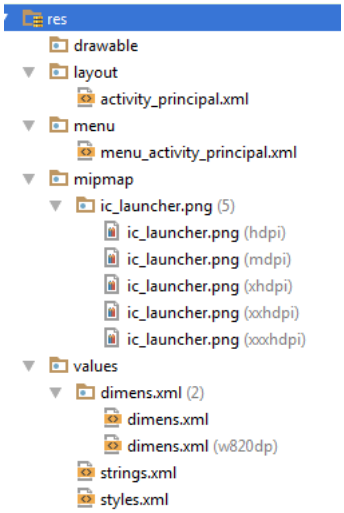
També podem aplicar estils diferents a l'atribut "theme". Si ens hi fixem, les "@" indiquen una carpeta del nostre projecte. Son els links als recursos on trobarà l'atribut concret. Per exemple: "@mipmap/ic_launcher" indica que l'atribut "icon" agafarà la imatge amb nom "ic_launcher" de la carpeta "mipmap".

```
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
    android:name=".ActivityPrincipal"
    android:label="@string/labelPrincipal" >
```

A la carpeta Java trobarem com a mínim una classe Java per cada pantalla (activity) que tingui la nostre aplicació. També podem generar més arxius Java si volem codi que no té res a veure amb una activitat en si, per exemple una classe ".java" que serveixi de Base de Dades local. Per defecte el programa ens genera la classe MainActivity que és el codi de l'activitat principal, tot i que podem modificar el nom:



Finalment tenim la carpeta recursos (res). Aquesta carpeta guarda tots els arxius XML que utilitzarem per la nostre aplicació, així com les imatges i les variables de text. Aquesta carpeta té varies subcarpetes per dividir els arxius XML i diferenciar-los per saber a què s'apliquen:

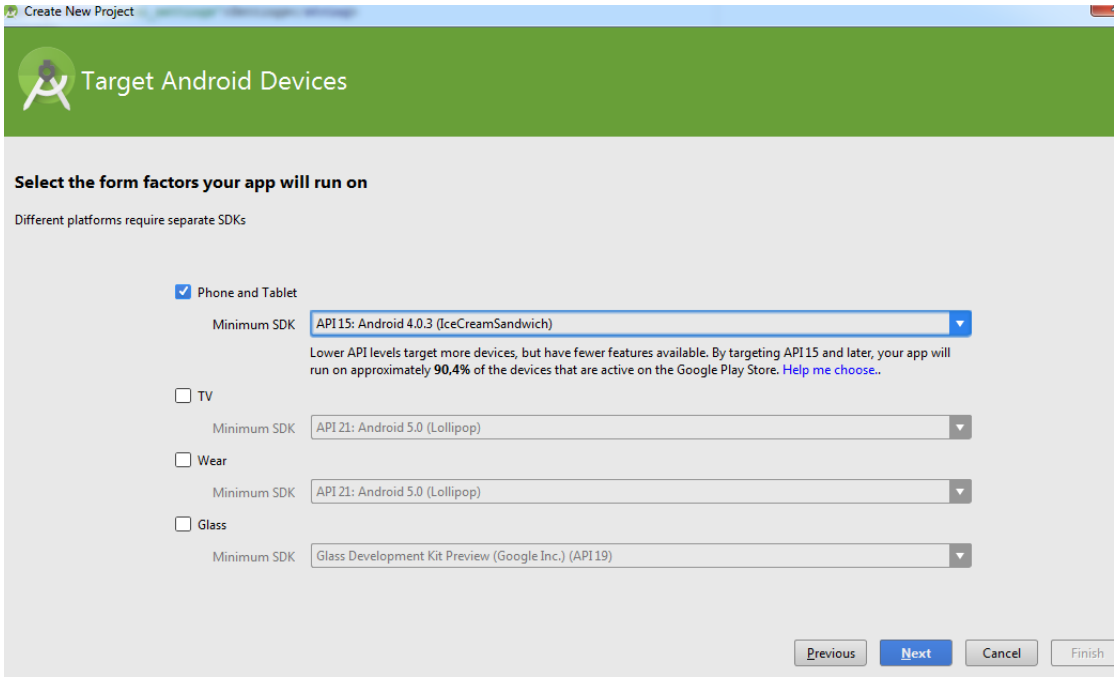


Aquests són els arxius que es generen per defecte al crear un projecte. Tenim un logo amb diferents mides per adaptar-se a les pantalles de tots els dispositius i els documents XML que formen part del layout, menú, cadenes de text (strings), dimensions i estils.

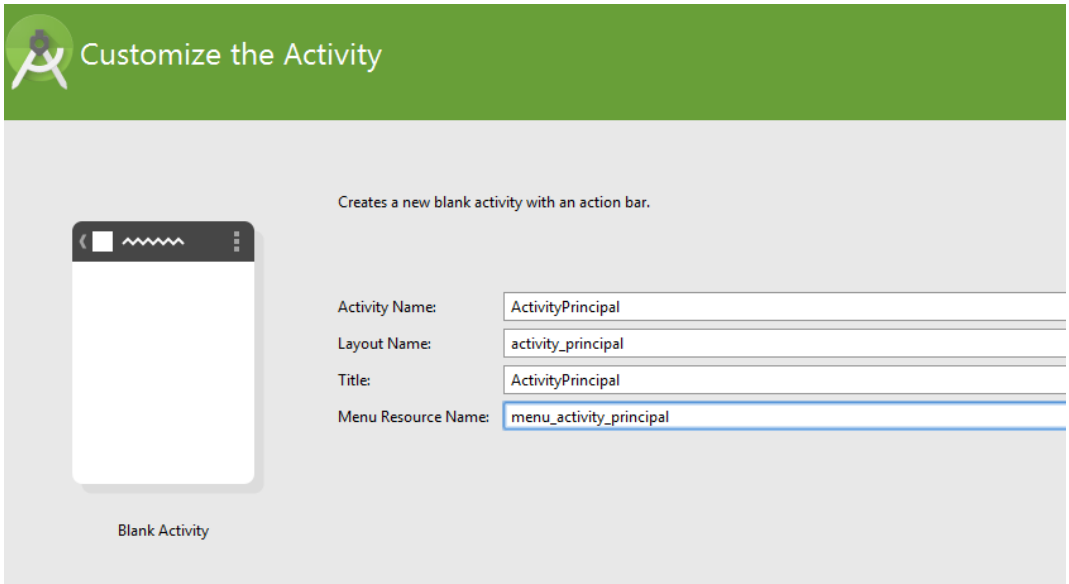
Cada cop que generem una nova activitat, necessitem un arxiu XML com a Layout per posar-hi els elements i una classe Java nova per programar-los i poder arrancar-la. Si volem que tingui menú, també es necessita un XML de menú per cada una.

5.3 Creació d'una activitat

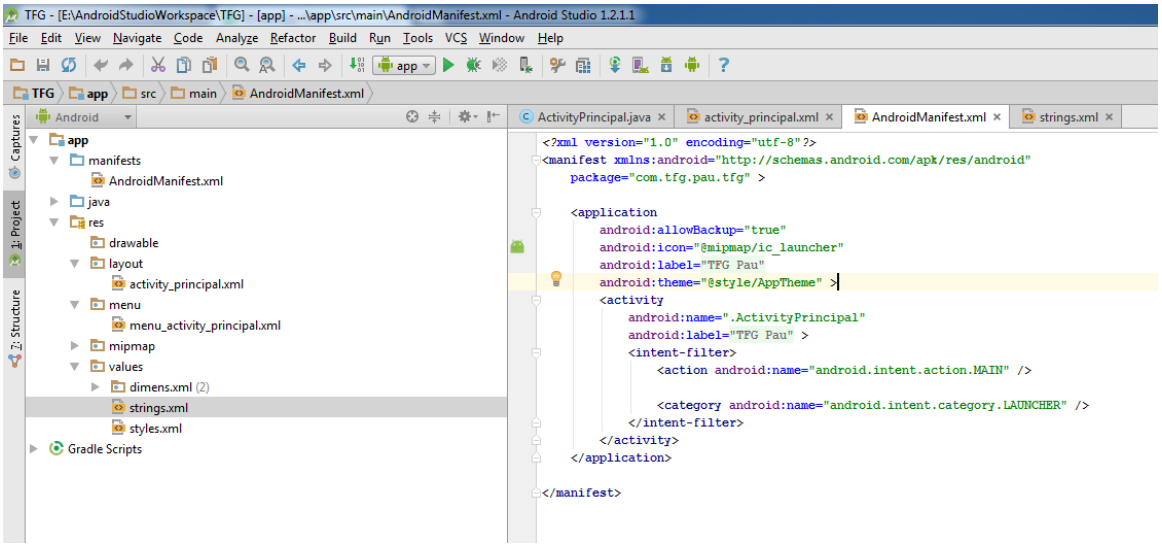
El primer de tot serà crear un nou projecte. Tots els passos els escollirem per defecte i posarem el nom que vulguem. Una de les coses importants serà el nostre target:



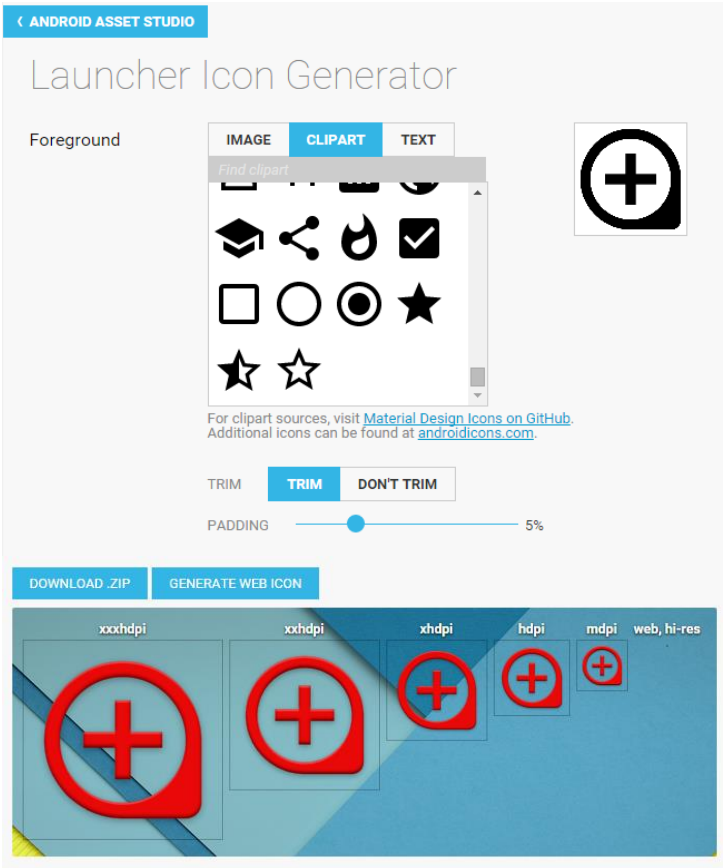
Utilitzarem l'API 15 (versió 4.0) amb la versió 4.0.3 que és la més actualitzada. La imatge ja indica que en aquesta versió el 90,4 % dels dispositius actius a Google Play podran executar-la sense problemes. Acte seguit, posarem el nom de l'activitat principal (arxius Java) i el seu layout (arxiu XML), amb el títol que vulguem per aquesta pantalla i el nom del arxiu XML del menú:



Seguint aquests passos, se'ns crearà el projecte:

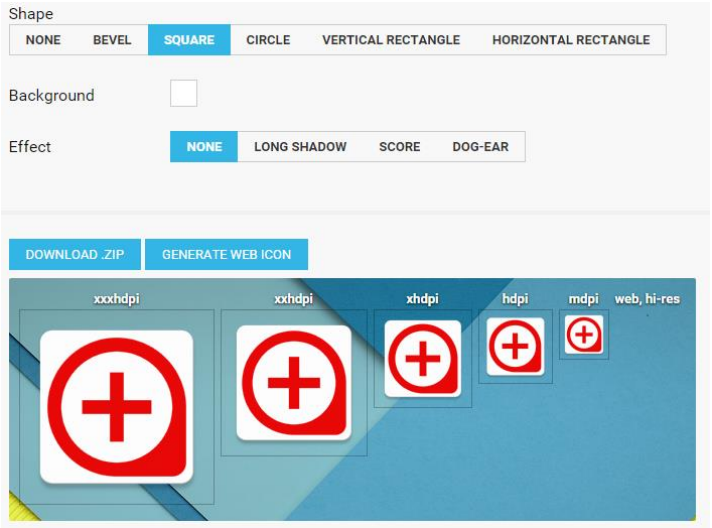


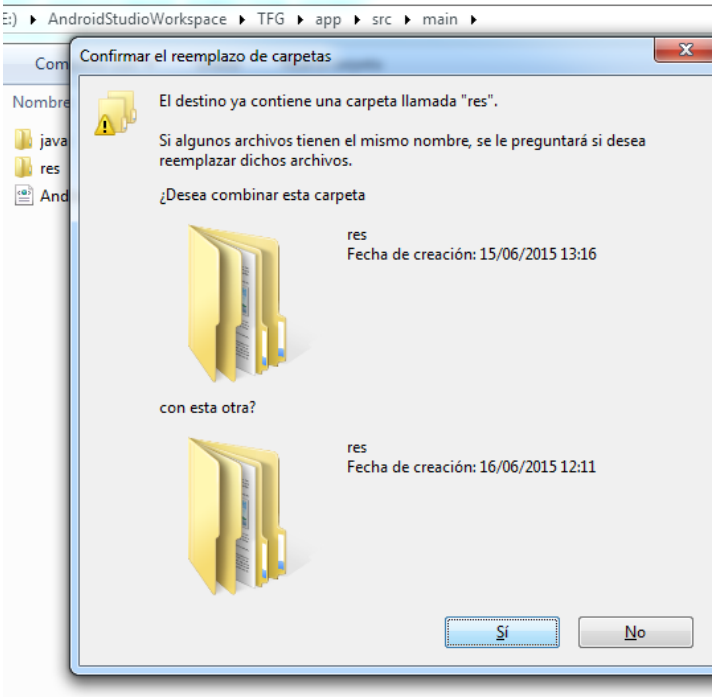
Per començar, canviarem el logo* de l'aplicació. Per fer-ho, utilitzarem una eina de google per generar icones de diferents mides per adaptar-se a les pantalles dels diferents dispositius. Es tracta d'una eina online:



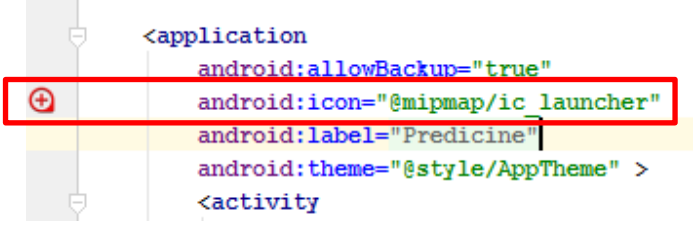
*Nota: Aquest no és el logotip definitiu, és un exemple per ensenyar aquesta eina.

Si posem diferents opcions també podem obtenir resultats diferents, com la forma "Square" amb fons blanc:



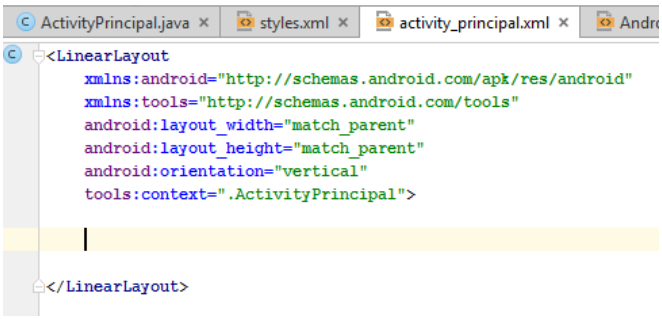


Un cop tenim un resultat que ens agradi, ens descarreguem el .zip i obtindrem una carpeta de recursos (res) amb la imatge en diferents mides. Haurem de buscar la ruta del nostre projecte per combinar aquesta carpeta "res" amb la de l'aplicació. D'aquesta manera mantindrem els arxius de la carpeta recursos del nostre projecte i afegirem les noves imatges. El següent pas serà indicar al Manifest aquest canvi i posarem el nom de les imatges al icona de l'aplicació.



El nom per defecte de les imatges que ens hem descarregat també és “ic_launcher” i per tant, el canvi s’haurà realitzat sol sense necessitat de variar el nom en el “AndroidManifest.xml”.

Ara ens dedicarem a maquetar els primers passos del nostre Mockup. Per fer-ho, anirem al layout de l’activitat principal que se’ns ha generat al crear el projecte. Tenim dos formes de veure el layout: la vista disseny i la vista text.



La vista text és el codi XML de l’estructura de l’aplicació. En una aplicació podem tenir elements (views) i layouts a mode de contenidors. Els dos casos es tracten com elements XML. Recordem com és l’estructura d’un element a XML:

<institut>
<aula>101</aula>
<lavabo planta=”1” />
</institut>

L’element arrel d’aquest exemple és institut. Conté dos elements: aula i lavabo, dels quals lavabo és un element buit que té un atribut anomenat planta. Aula és un element sense atributs però no està buit, conté el 101.

En Android tractem els layouts com elements que tenen els atributs que vulguem. També poden contenir altres elements, ja siguin views o altres layouts. En canvi, els views són elements buits que no contenen altres elements i/o text. El més important ens els views i els layouts son els atributs que marquen totes les característiques que tindran.

Hi ha diferents tipus de layouts: Linear, Relative, Table, Grid i Frame. Els que més s’utilitzen son el LinearLayout i el RelativeLayout perquè els altres son per casos més concrets de taules i estructures que també es poden crear utilitzant aquests dos. En el meu cas he utilitzat un LinearLayout amb els atributs bàsics que ha detenir com a Layout “pare”. Aquest layout col·loca els elements que hi posem dins seu de forma linear i li hem d’indicar si els volem de forma vertical u horitzontal:

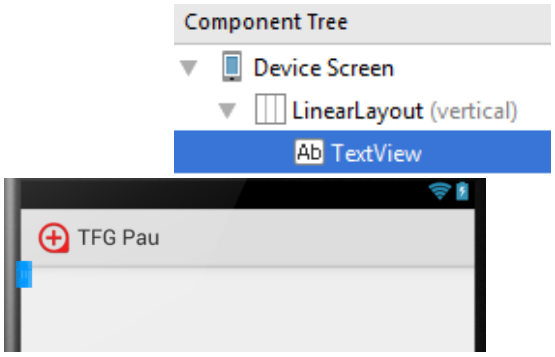
```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
```

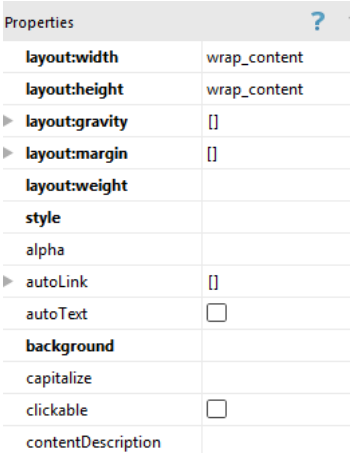
Els atributs “width” i “height” son la llargada i l’amplada. “Match parent” indica que agafarà les mides de l’element pare, en aquest cas de la pantalla sencera. D’aquesta manera tenim un layout que fa de contenidor amb la mida de tota la pantalla. Ara podem començar a posar-hi elements:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ActivityPrincipal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

Un TextView és un element buit (abans hem dit que els “views” son elements buits) que mostra un text amb les característiques que vulguem. Aquestes característiques les podem escriure aquí a mà, com hem fet amb l’alçada i l’amplada (“wrap_content” significa que s’adaptarà al contingut que posem al view), o podem modificar-les a les propietats de la vista “Design”.

A la vista de disseny també veiem una previsualització de com queda el que estem fent a un mòbil, un arbre de components i les propietats de l’element seleccionat:

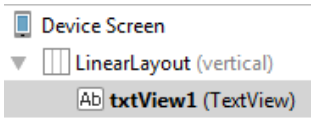




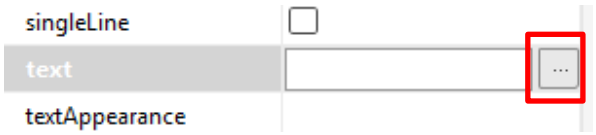
A les propietats tenim un scroll vertical perquè hi ha una llista llarga. També varien algunes depenent del tipus d’element. Per

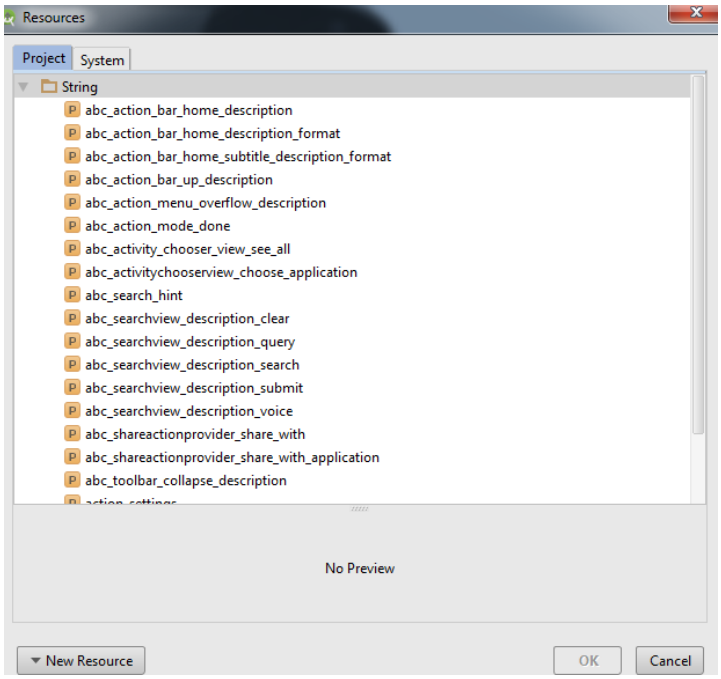
exemple, busquem la propietat ID i li assignem un nom diferent des de aquesta vista:

Podem observar a l'arbre de components que ha canviat el nom:

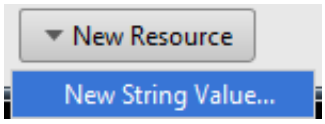


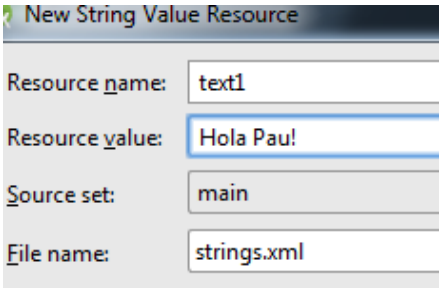
Per mostrar text, anem a la propietat "text" i allà podrem escriure el que vulguem. En comptes d'escriure un text directament, el que farem serà assignar-li una variable tipus String per poder controlar-lo millor per codi. Podem escriure un text directament és clar, però ens costarà més de gestionar quan programem en Java. Per fer-ho, seleccionem el menú de la propietat text i se'ns mostra aquesta finestra:





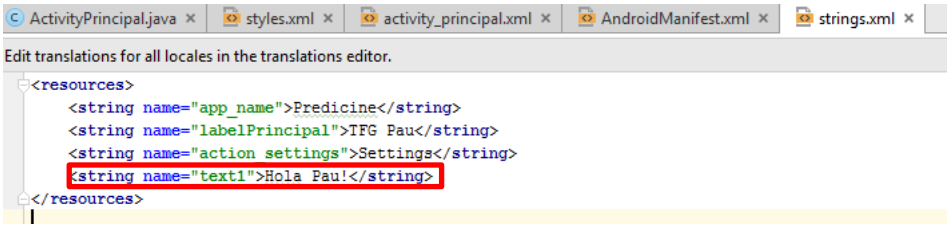
Crearem un nou recurs d'String que servirà de variable per aquest text:





D'aquesta manera ens crearà una variable anomenada "text" amb el valor per defecte que li hem posat, en aquest cas "Hola Pau!". A la propietat del text, veurem la ruta on anirà a buscar aquesta cadena de caràcters per mostrar-la.

Si comprovem el fitxer "Strings.xml" de la carpeta recursos (res), veurem que s'ha generat sol. També podríem editar aquest fitxer primer per crear una variable de text nova i després assignar la ruta "@string/nom_variable" a la propietat "text" de qualsevol element.



Aquestes variables tenen format XML; son elements amb el contingut de la variable que vulguem i amb l'atribut "name" per assignar el nom de la variable.

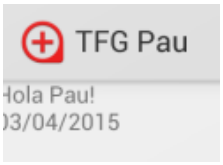
Ara podem canviar de vista un altre cop per mostrar el text d'aquest TextView que hem modificat:

Observem que els atributs “id” i “text” els ha escrit automàticament quan els hem editat des de la vista de disseny. Per tant, hem vist dos maneres d’afegir atributs als nostres elements. Seguim ara creant nous elements de la nostre pantalla, com la data. Necessitarem un altre TextView:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/txtView2"
    android:text="@string/text2" />
```

Recordem que haurem de crear la variable “text2” al fitxer “Strings.xml” amb la data que volem posar. Serà més fàcil canviar la data quan passi un dia si fem referència al recurs “@string/text2” mitjançant codi Java, però això ja ho veurem.

Utilitzant només aquestes propietats, el resultat queda d’aquesta manera:



Nosaltres el que volem és col·locar la data a la mateixa altura que l’altre text. També volem que el text sigui de color negre i més gran. Hi ha varies maneres de fer-ho, creant més layouts i tocant propietats. Així ha quedat després de fer-ho:

ctivityPrincipal.java × | styles.xml × | activity_principal.xml × | Andr

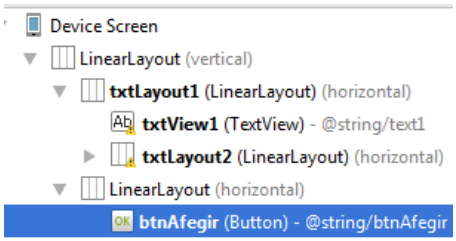
```
:LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ActivityPrincipal">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:id="@+id/txtLayout1"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:layout_marginTop="25dp"
        android:layout_marginBottom="5dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txtView1"
        android:text="Hola Pau!"
        android:textSize="20dp"
        android:textColor="#ff000000" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:id="@+id/txtLayout2"
            android:gravity="right">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txtView2"
        android:text="03/04/2015"
        android:textSize="20dp"
        android:textColor="#ff000000" />

        </LinearLayout>
    </LinearLayout>

:/LinearLayout>
```

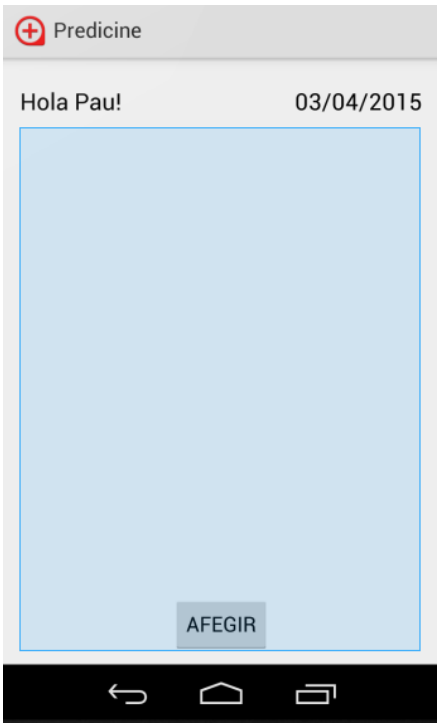
Fixem-nos en els atributs del textView2 i dels nous LinearLayouts que he creat. Aquest contenidor nou l’utilitzo per indicar que vull disposar els dos textsViews en forma horitzontal i

l'altre per indicar que el text dins seu (el textView2) es situï a la dreta amb la propietat gravity="right".

Per crear un botó, seguirem els mateixos passos. El botó és un View amb algunes propietats diferents als dels texts, però no canvien massa.

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="bottom|center_horizontal"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="5dp"
    android:layout_marginBottom="10dp">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/btnAfegir"
        android:id="@+id/btnAfegir" />
</LinearLayout>
```

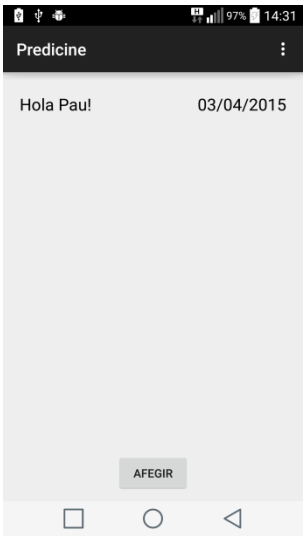


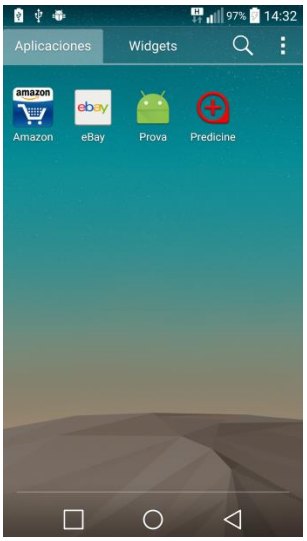


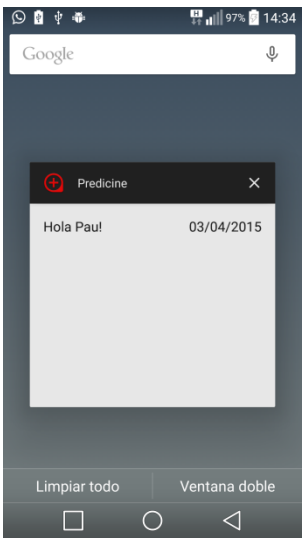
Hem creat un altre LinearLayout que també farem servir per contenir el futur ListView que posarem per llistar els medicaments. El resultat fins ara ha quedat així, fem una ullada al fitxer "Strings.xml":

```
<resources>
    <string name="app_name">Predicine</string>
    <string name="labelPrincipal">Predicine</string>
    <string name="action_settings">Settings</string>
    <string name="text1">Hola Pau!</string>
    <string name="text2">03/04/2015</string>
    <string name="btnAfegir">AFEGIR</string>
</resources>
```

Hem d'anar seguint fent aquests passos per maquetar l'aplicació i totes les seves pantalles i activitats. Veiem com queda el resultat executat amb captures de pantalla del meu mòbil:







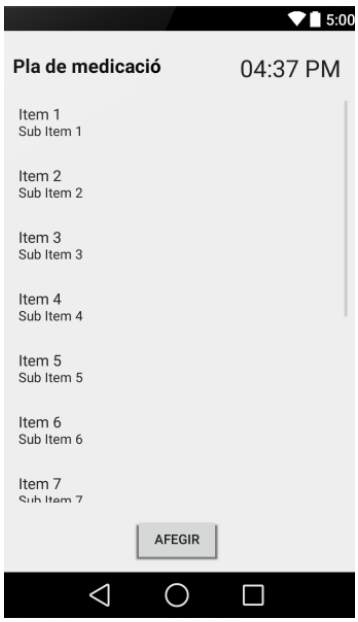
5.4 Maquetació

El desenvolupament de la meva aplicació requerirà la creació de quatre activitats Android. Cada activitat, com ja hem vist, necessita ser declarada al "AndroidManifest.xml" i ha de tindre un layout en XML i una classe de Java.

Tot seguit explicaré els passos que he seguit per maquetar i programar cada una de les activitats.

Cada activitat és una pantalla de la meva aplicació:

-Activitat principal



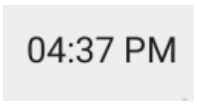
Es tracta de la pantalla principal. Aquí es mostra el pla de medicació, és a dir, els medicaments que ha afegit l'usuari en forma de llista dinàmica. L'usuari podrà afegir més medicaments al seu pla de medicació mitjançant un botó o podrà tocar amb el dit sobre qualsevol medicament per veure tota la informació que s'ha introduït prèviament, com la dosi i el número de càpsules restants. Aquesta informació es mostrarà en una altre activitat anomenada "InfoMedicament".

L'element de la llista és el més important d'aquesta activitat. Farem que s'adapti al "layout" contenidor en alçada i amplada amb un marge a tots els costats de 10 dp ("density pixels"). També li donarem un identificador únic amb el qual accedirem al objecte llista des de Java, en aquest cas "lv1". Aquest element és senzillament una espècie de contenidor per la llista, però al ser una llista dinàmica necessita altres elements per funcionar i s'utilitzen a la classe Java de l'activitat. També haurem d'indicar l'aspecte que tindrà un "ítem" de la llista per aplicar-lo a tots els medicaments que puguem crear dins la llista. En el meu cas he utilitzat un per defecte que porta el "sdk" d'Android perquè m'interessava que aparegués el nom del medicament amb l'estructura que veurem a la part de programació.

```
<ListView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_margin="10dp"
    android:id="@+id/lv1"
    android:layout_gravity="center_horizontal"
    android:layout_weight="1" />
```

La hora és un element que es diu "TextClock" i agafa l'hora del sistema per mostrar-la. No és un element gaire important perquè gairebé tots els mòbils mostren la hora però facilita aquesta informació que és essencial per prendre medicaments.

```
<TextClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textClock"
    android:textSize="25dp" />
```



*Nota: Aquest element requereix una versió d'Android mínima, la API 17, per això està subratllat en vermell. Qualsevol dispositiu amb una versió Android més antiga no podrà visualitzar el rellotge.

Els altres elements són més típics. Un "LinearLayout" amb orientació vertical fa d'element contenidor i de "pare" a la jerarquia XML. El text i el botó són elements simples i ens assegurarem de posar un identificador a cada un d'ells per després accedir-hi des de la classe Java de l'activitat.

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="AFEGIR"
    android:id="@+id/btnAfegir" />
```

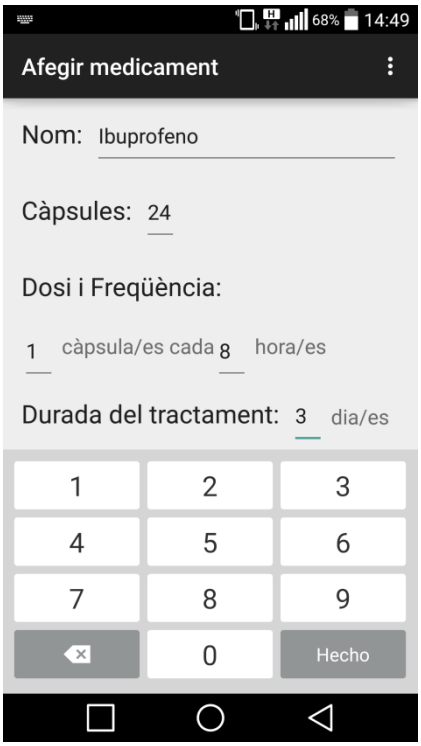
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/txtView1"
    android:text="@string/text1"
    android:textSize="20dp"
    android:textColor="#ff000000"
    android:textStyle="bold" />
```

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ActivityPrincipal"
    android:weightSum="1">
```

El LinearLayout és un element XML que no està buit perquè conté els altres elements a dins. Per tant, podem veure al requadre vermell que és un element XML sense tancar a diferència dels altres en blau, perquè es tanca al final de tot:

```
</LinearLayout>
```

-Activitat per afegir un medicament



Aquesta és la pantalla que permet afegir un medicament al pla de medicació de l'usuari mitjançant un formulari. El nom de l'activitat és "AfegirMedicament". Un botó permet guardar les dades i afegir el medicament i un altre tornar al pla de medicació sense canvis. Tots els camps del formulari són obligatoris i no es permet afegir el medicament deixant un camp buit. Es guarden totes les dades en camps únics per poder accedir a elles més tard mitjançant les altres activitats. Un cop afegit el medicament, es torna automàticament a la pantalla principal i es mostra a la llista el nom del medicament afegit.

Els elements d'aquesta pantalla són elements bàsics. A diferència de l'activitat principal, aquí tenim cinc editors de text. XML permet donar diferents propietats a cada un d'ells per adaptar-los a diferents necessitats.

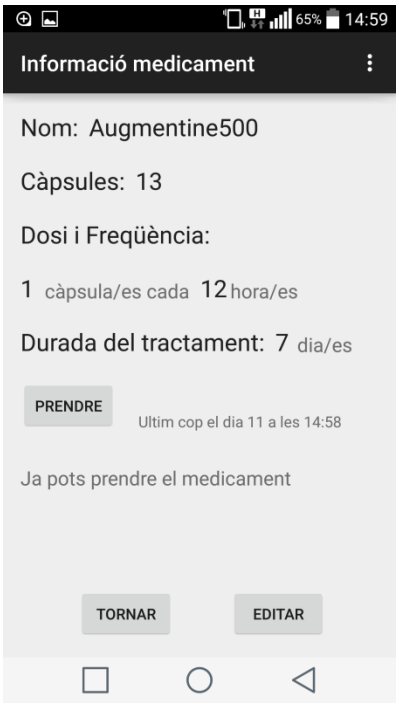
```
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/et1"
    android:layout_alignParentTop="true"
    android:layout_toRightOf="@+id/textView"
    android:layout_marginLeft="10dp" />
```

Aquest és l'editor de text on l'usuari introduirà el nom del medicament. A diferència del altres, aquest s'adapta a l'amplada de l'element arrel o pare (Layout) i per això és més ample que els altres. D'aquesta manera l'usuari pot veure el que escriu sense problemes.

```
<EditText
    android:layout_width="30dp"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/et2"
    android:layout_alignTop="@+id/textView2"
    android:layout_toRightOf="@+id/textView2"
    android:layout_toEndOf="@+id/textView2"
    android:layout_marginLeft="10dp" />
```

Els altres quatre editors de text són idèntics pràcticament, canviant només la seva posició. La diferència és l'atribut "input type number". Quan l'usuari toqui l'editor de text, en comptes d'aparèixer un teclat virtual normal a la pantalla del seu mòbil, li apareixerà un teclat només de números. D'aquesta manera podem controlar que només es poden entrar números sense fer servir codi Java, només amb un atribut de l'element "EditText" en XML.

-Activitat per mostrar la informació del medicament



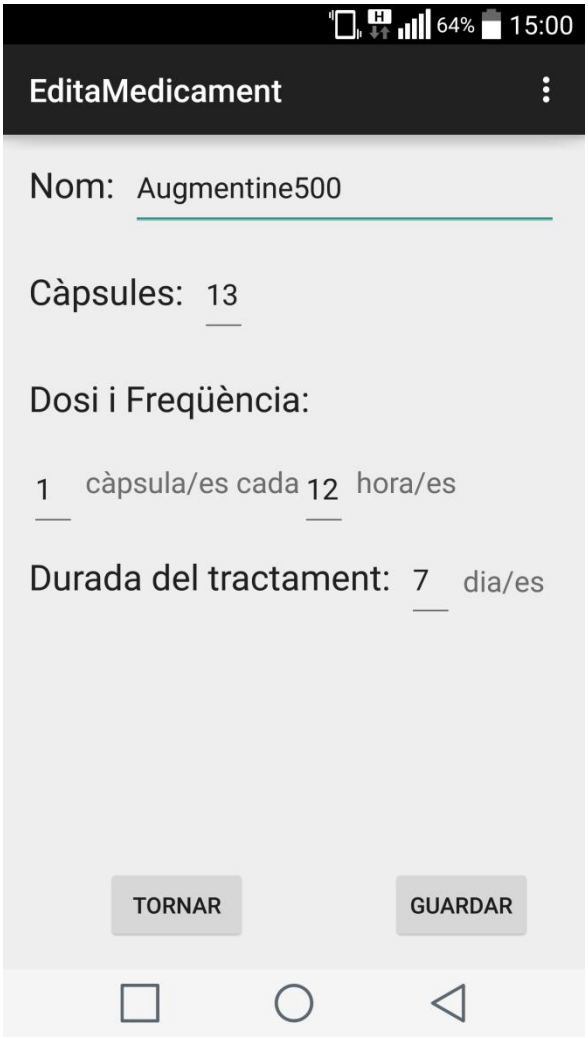
Tots els medicaments que ha afegit l'usuari es mostren a la llista de l'activitat principal. Tocant un dels elements de la llista (un medicament) es podrà accedir a tota la seva informació. D'aquesta manera se'ns obrirà l'activitat "InfoMedicament" que dèiem anteriorment. Ens mostrarà les dades que es van introduir a la pantalla d'afegir-lo. Des de aquesta pantalla podrem realitzar tres accions mitjançant tres botons diferents: podrem tornar a la pantalla principal, podrem editar la informació sempre que vulguem mitjançant una nova activitat i finalment podrem prendre el medicament. Aquesta última acció resta una càpsula a la variable "càpsules restants" i mostra

dos textos nous. Un indica el dia i l'hora de l'última vegada que l'usuari ha pres aquell medicament i l'altre mostra un temporitzador per saber quan es pot tornar a prendre. El temporitzador pren la variable de la freqüència amb la que s'ha de prendre aquell medicament i desactiva el botó "Prendre" fins que no ha acabat. Una notificació avisa a l'usuari que ja pot prendre el medicament en qüestió quan el temporitzador arriba a zero. Una altre notificació també alerta a l'usuari en cas que li quedin tres càpsules o menys per avisar-lo que s'està quedant sense. Tot això ho explicaré amb més detall a la part de programació.

No hi ha elements a destacar en aquesta activitat, tot son "TextViews" i "Buttons" controlats mitjançant el seu identificador a través de Java per mostrar el seu contingut o per realitzar alguna acció.

-Activitat per editar el medicament

Aquesta activitat és similar a la d'afegir un medicament però en aquest cas els camps son els d'un medicament ja existent. L'usuari podrà variar la seva informació i es sobreescriurà correctament al guardar els canvis. Un cop s'ha tocat el botó de guardar canvis, es torna a la pantalla principal. També es pot tocar el botó de tornar enrere sense canvis i, en aquest cas, es torna a l'activitat de "InfoMedicament" que és on està el botó per accedir a la pantalla d'edició.

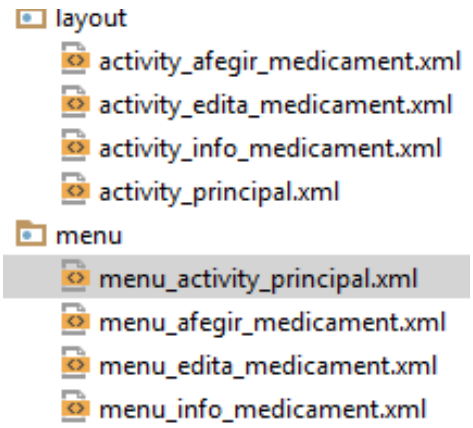


El text que apareix als editors de textos estan generats per codi perquè agafa les variables que li envia l'activitat "InfoMedicament". Aquesta pantalla juntament amb la d'informació, són modificades en funció del medicament que hem escollit de la llista principal. L'estat dels editors de textos en l'XML original és diferent del que veurem sempre, ja que en realitat no tenen text però el generem a través de Java com veurem més endavant.

```
<EditText
    android:layout_width="30dp"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/et5"
    android:layout_alignTop="@+id/textView6"
    android:layout_toRightOf="@+id/textView6"
    android:layout_toEndOf="@+id/textView6"
    android:layout_marginLeft="10dp" />
```

Podem comprovar com aquest editor de text no té ninguna propietat “android:text” en el seu estat original.

Un cop vist l’estructura bàsica de les quatre pantalles anem a mirar els seus menús. De fet, l’estructura dels menús és igual per les quatre activitats:



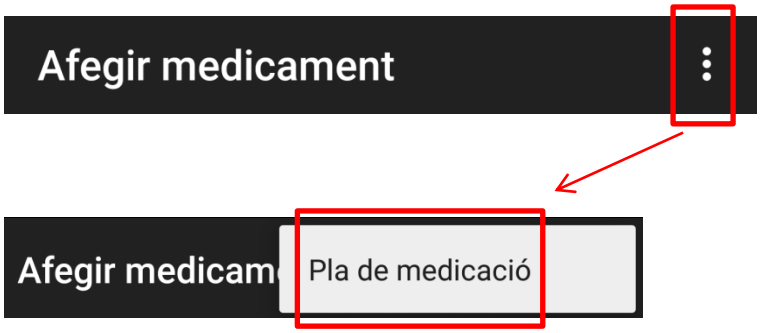
-Menú

El menú es genera automàticament al crear una activitat amb Android Studio. L’estructura del menú és una mica diferent dels layouts de les activitats. El menú es troba a la barra superior de cada activitat i per defecte, et genera un “item” anomenat “Settings”. En el meu cas, el menú de cada pantalla t’ofereix una opció per tornar a la pantalla principal.

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.tfg.pau.tfg.InfoMedicament">

    <item
        android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />
</menu>
```

Aquest menú és el mateix per la resta d’activitats. Veiem que l’element arrel és un menú que pot tenir elements dintre, els items. Per cada acció que vulguem al menú

haurem de generar un nou item amb un identificador únic igual que la resta d'elements per poder accedir-hi per codi.



Fins ara hem vist els "layouts" de les quatre activitats i l'estructura bàsica dels seus menús, però fem una ullada al resultat final del manifest i el fitxer de recursos de cadenes de text.

-Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tfg.pau.tfg" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".ActivityPrincipal"
            android:label="@string/labelPrincipal" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".AfegirMedicament"
            android:label="@string/title_activity_afegir_medicament" >
        </activity>
        <activity
            android:name=".InfoMedicament"
            android:label="@string/title_activity_info_medicament" >
        </activity>
        <activity
            android:name=".EditaMedicament"
            android:label="@string/title_activity_edita_medicament" >
        </activity>
    </application>

</manifest>
```

Podem veure que les quatre activitats estan declarades correctament. Es recull d'aquest document el títol de cada activitat, així com el nom de l'aplicació i el seu logotip.

-Strings.xml

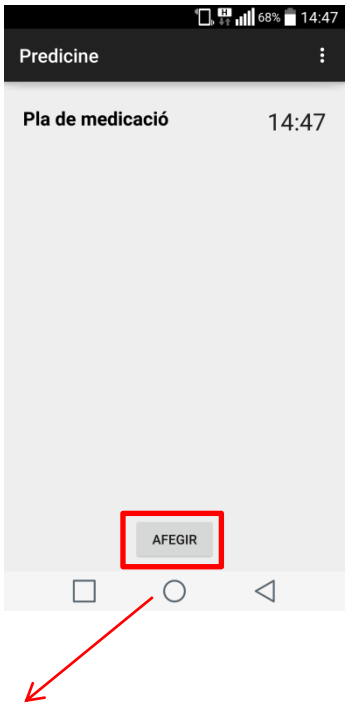
```
<resources>
    <string name="app_name">Pillbox</string>
    <string name="labelPrincipal">Pillbox</string>
    <string name="action_settings">Pla de medicació</string>
    <string name="text1">Pla de medicació</string>
    <string name="btnAfegir">Afegir</string>
    <string name="title_activity_afegir_medicament">Afegir medicament</string>
    <string name="afegirtxt1">Nom:</string>
    <string name="afegirtxt2">Càpsules restants:</string>
    <string name="afegirtxt3">Dosi i Freqüència:</string>
    <string name="afegirtxt4">càpsula/es cada</string>
    <string name="afegirtxthores">hora/es</string>
    <string name="afegirtxt6">Durada tractament:</string>
    <string name="afegirtxtdies">dia/es</string>
    <string name="btnTornar">Tornar</string>
    <string name="title_activity_info_medicament">Informació medicament</string>
    <string name="btnprendre">Prendre</string>
    <string name="btnedita">Editar</string>
    <string name="title_activity_edita_medicament">EditaMedicament</string>
    <string name="btnGuardar">Guardar</string>

</resources>
```

Aquests són tots els textos que no es generen per codi, com per exemple el títol de les activitats o els TextViews dels formularis. Les quatre activitats, els seus menús i el AndroidManifest utilitzen aquest fitxer de recursos per posar text als seus elements i variables.

5.5 Programació

Tot el projecte Android Studio l'entregaré juntament amb la memòria per poder veure tot el codi Java i altres documents del projecte que ha anat comentant fins ara. En aquest apartat, explicaré les funcionalitats de l'aplicació mitjançant captures de pantalla i després ensenyaré la part del codi que la permet funcionar.

5.5.1 Activitats/pantalles



Aquesta és l'activitat principal sense medicaments afegits a la llista. Els textos que apareixen són estàtics i no cal modificar-los per codi.

Quan l'usuari toca el botó, arranquem l'activitat AfegirMedicament. Per fer-ho, hem de crear un objecte "Button" i que busqui la "id" del botó maquetat per afegir:

```
private Button bt;

bt = (Button) findViewById(R.id.btnAfegir);
```

El mètode "findViewById" busca el nom del identificador que introduïm als elements del layout vinculat al codi de l'activitat. Per exemple, encara que tinguéssim un botó amb el mateix ID "btnAfegir" al layout de "InfoMedicament", no el detectaria perquè no és el layout de la classe Java que estem modificant.

```
bt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent int1 = new Intent(ActivityPrincipal.this, AfegirMedicament.class);
        contador = strArr.size();
        String scontador = Integer.toString(contador);
        int1.putExtra("cont", scontador);
        startActivityForResult(int1, code);
    }
});
```

Li assignem un OnClickListener al objecte botó que hem creat a partir de la ID btnAfegir. Aquest mètode es va executant per detectar si l'usuari toca el botó a la pantalla. Un cop ho fa s'executa el mètode onClick i tot el codi que conté.

Un "intent" és una traducció literal de "intenció". Quan volem llançar qualsevol acció com obrir l'aplicació càmera del mòbil o el teclat del telèfon per trucar, ho fem amb un intent. Ena quest cas, utilitzem un intent per cridar l'activitat "AfegirMedicament.class". Un cop tenim assignat l'intent (int1), podem enviar dades a la activitat que llançarem amb el mètode "putExtra". Té un funcionament simple, envia una dada en forma de variable-valor (en el nostre cas, variable "cont" i el valor es el contingut de la variable "scontador", una cadena de text que conté la mida del Array que controla la llista de medicaments ("strArr" és el nom de la variable StringArray: per saber la mida utilitzem "strArr.size()"). Bàsicament, aquesta dada l'enviem per avisar a la pantalla d'afegir medicament sobre quants medicaments tenim actualment i així l'insereixi al final de la llista sempre.

Per iniciar una nova activitat, tenim dos opcions:

-Mètode startActivity(intent):

Aquest mètode és el més simple i arranca la pantalla de l'activitat que li hem passat a la variable intent. Per poder tornar a la primera activitat ho hauríem de fer amb un altre intent de la mateixa manera.

-Mètode startActivityForResult(intent,code):

Aquest mètode és mes complex que l'anterior. Serveix per arrancar la nova activitat però a la espera d'una resposta. En altres paraules, la nova activitat ha de donar un resultat i després es torna a l'activitat que l'ha cridat, la principal en el nostre cas. Aquest mètode necessita un altre mètode a l'activitat principal: "onActivityResult". És un mètode que s'executa quan hem acabat les nostres funcions a la segona activitat i tornem a la primera. Es necessita un codi per saber quina és l'activitat que ha finalitzat en cas de tenir moltes activitats llançades amb aquest mètode, perquè sinó

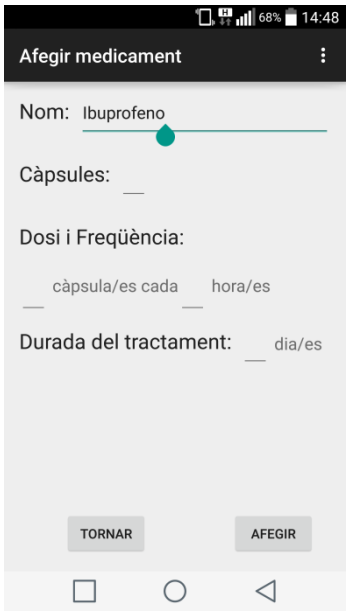
s'executaria sempre onActivityResult en tots els casos. És una manera de filtrar les accions en aquest mètode.

Tornant a la meva activitat, utilitzo un startActivityForResult i li envio "code" declarat al inici:

```
int code=1234,code1=1235;
```

La variable "code1" l'utilitzaré més endavant.

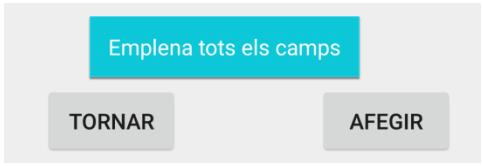
A partir d'aquí, l'aplicació executa la classe AfegirMedicament i es manté a l'espera d'una resposta. La resposta pot ser "RESULT_OK" i "RESULT_CANCELED" i per tornar a l'activitat principal utilitzarem "finish()".



Aquesta activitat té dos opcions: Afegir el medicament o tornar enrere. Si tornem, es crida el mètode onClick del botó i assignem RESULT_CANCELED a l'activitat i l'acabem amb finish().

```
bt2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();
        setResult(RESULT_CANCELED, intent);
        finish();
    }
});
```

Si afegim el medicament, primer es comprova que tots els camps estiguin plens. En cas contrari, no deixa afegir i es mostra un missatge informatiu:

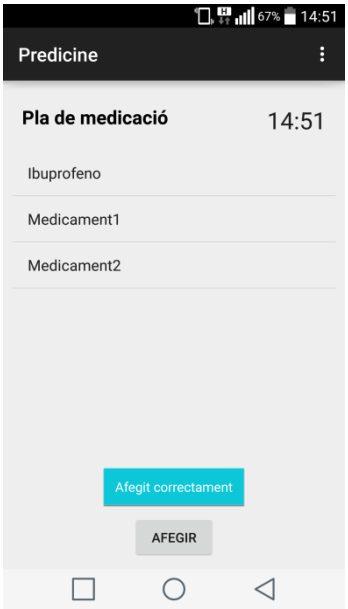



```
if (et1.getText().toString().equals("")||et2.getText().toString().equals("")
        ||et3.getText().toString().equals("")||et4.getText().toString().equals("")
        ||et5.getText().toString().equals("")) {
    Toast.makeText(getApplicationContext(), "Emplena tots els camps", Toast.LENGTH_SHORT).show();
```

Aquests missatges es diuen “Toast” i son simples d’utilitzar, tal i com es veu al requadre vermell.

Si tot està correcte, s’afegirà el medicament a la llista de la pantalla principal, s’acabarà aquesta activitat i tornarem un altre cop:

```
else {
  Intent intent = new Intent();
  String txt1=et1.getText().toString();
  String txt2=et2.getText().toString();
  String txt3=et3.getText().toString();
  String txt4=et4.getText().toString();
  String txt5=et5.getText().toString();
  SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
  SharedPreferences.Editor editor = prefs.edit();
  editor.putString("nom"+contador, txt1);
  editor.putString("capsules"+contador, txt2);
  editor.putString("dosi"+contador, txt3);
  editor.putString("frequencia"+contador, txt4);
  editor.putString("durada"+ contador, txt5);
  editor.commit();
  setResult(RESULT_OK, intent);
  finish();
```



Declarem “Strings” (cadenes de text) amb els valors que contenen els editors de text. Passem aquests valors a Strings amb el mètode “toString()” perquè la majoria de valors són enters. Per guardar els valors ho farem mitjançant “SharedPreferences”. És un document XML que es guarda al mòbil i només es pot accedir a ell si som usuaris “root”. És un document semblant al de Strings.xml que guarda en format variable-valor les dades i es guarda encara que l’aplicació es tanqui. Fem servir un editor per modificar o crear variables en aquest document SharedPreferences mitjançant el mètode “editor.putString(“variable”,valor)”. El contador és una variable que indica la posició de la llista on es trobarà el medicament. És la variable que li hem passat anteriorment amb el putExtra() a l’activitat principal. D’aquesta manera podem crear SharedPreferences diferents per cada medicament. Per exemple, si no hi ha cap element a la llista, contador valdrà 1 i es guardaran a SharedPreferences els valors “nom1, capsules1, dosi1...”. Quan afegim un segon medicament, contador valdrà 2 i guardarà les dades amb noms diferents “nom2, capsules2...”. Per això afegim contador al final del nom de la variable:

```
editor.putString("nom"+contador, txt1);
```

SharedPreferences ens permet guardar moltes dades de forma dinàmica utilitzant aquest sistema i així ens estalviem la creació d'una base de dades petita d'una sola taula. Les bases de dades Android utilitzen la classe SQLite, que és senzilla d'utilitzar però consumeix molts recursos.

Quan acabem d'editar les SharedPreferences, utilitzem "editor.commit()" i assignem el resultat OK de l'activitat i l'acabem amb finish().

A l'activitat principal, s'executa el mètode onActivityResult que dèiem quan es retorna de AfegirMedicament:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    //comprovem que el resultat ha sigut "ok"
    if(requestCode==1234 && resultCode==RESULT_OK){
        //recupero la informació de la activitat on l'usuari ha creat el medicament
        int valor = strArr.size();
        SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
        String txt1 = prefs.getString("nom"+valor, "default");
        String txt2 = prefs.getString("capsules"+valor, "default");
        String txt3 = prefs.getString("dosi"+valor, "default");
        String txt4 = prefs.getString("frequencia"+valor, "default");
        String txt5 = prefs.getString("durada"+valor, "default");
        strArr.add(txt1);
        adapter.notifyDataSetChanged();
        Toast.makeText(getApplicationContext(), "Afegit correctament", Toast.LENGTH_SHORT).show();
    }
    /*si obté resultat "canceled", indiquem amb un toast que no fem canvis. Això passa al clicar el botó cancel·lar de la
     segona activitat o amb el botó retornar del mòbil*/
    if(requestCode==1234 && resultCode==RESULT_CANCELED){
        Toast.makeText(getApplicationContext(), "S'ha tornat sense canvis", Toast.LENGTH_SHORT).show();
    }
```

Primer de tot comprovem amb la condició si el codi és el que havíem enviat abans per saber si onActivityResult ha estat cridat per AfegirMedicament o no. Si el codi és correcte i el resultat és RESULT_CANCELED, mostrem un missatge conforme s'ha tornat sense canvis perquè això vol dir que l'usuari ha tocat el botó de tornar o el de retorn del propi mòbil. Si és RESULT_OK, recuperem les preferències de SharedPreferences amb el mètode "prefs.getString(variable-valor)". Aquest mètode ha de tenir un valor per defecte per si la variable no existeix. Si no el programa fallaria si no controléssim l'excepció amb un "try-catch".

```
String txt1 = prefs.getString("nom"+valor, "default");
```

La variable "valor" és la posició de la llista que toca, així agafem les dades correctes si n'hi haguéssim més d'una. El valor "default" és el que agafaria si "nom+valor" no existeix. De tota manera sempre existirà perquè l'hem creat a l'activitat AfegirMedicament.

Ara només ens falta mostrar a l'usuari el nou medicament a la llista. Per fer-ho, afegirem el nom del medicament a una fila del Array i l'adaptador s'encarregarà de col·locar-lo correctament.

La llista dinàmica treballa a partir del ListView que hem maquetat a l'apartat corresponent. La llista també és un objecte i la declarem així:

```
private ListView lv;
private ArrayList<String> strArr;
private ArrayAdapter<String> adapter;
```

També declarem la llista Array aquí i l'adaptador per llistes dinàmiques.

```
lv = (ListView) findViewById(R.id.lv1);

strArr = new ArrayList<String>();
adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, strArr);
lv.setAdapter(adapter);
```

Al adaptador li assignem una estructura simple que la podem trobar a la llibreria SDK d'Android. Aquesta estructura es pot personalitzar mitjançant un nou recurs XML. Bàsciament aquest document descriu al adaptador com ha de ser un "ítem" de la llista per després duplicar-lo cada cop que es genera un ítem nou amb els noms diferents. Després, li diem també que l'adaptador és del ArrayList "strArr" i assignem al objecte ListView el adaptador que treballa amb el nostre ArrayList. D'aquesta manera, al afegir el nom del medicament mitjançant "strArr.add(txt1)" i notificant els canvis al adaptador amb "adapter.notifyDataSetChanged()", automàticament ens crea una nova fila de la llista amb el nom del medicament que acabem de afegir.

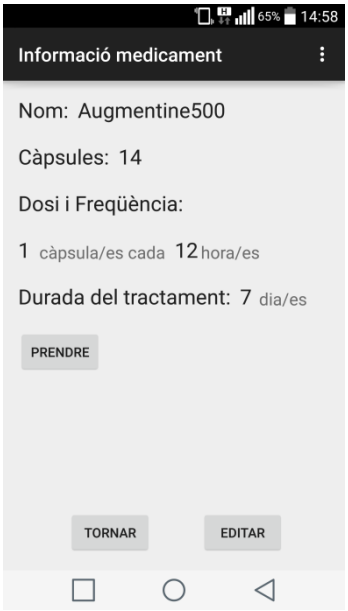
```
strArr.add(txt1);
adapter.notifyDataSetChanged();
```

Així funciona la manera d'afegir i actualitzar medicaments a la llista, gràcies al formulari d'AfegirMedicament. El següent pas és veure la informació de cada medicament per separat. Per fer-ho, necessitarem un mètode per saber si l'usuari toca un element de la llista o un altre i després arrancar l'activitat "InfoMedicament" enviant la posició de la llista que s'ha tocat.

```
lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Intent int2 = new Intent(ActivityPrincipal.this, InfoMedicament.class);
        infoint=position;
        int2.putExtra("pos",position);
        startActivityForResult(int2, code1);
    }
});
```

Utilitzem un sistema similar al de afegir medicament, simplement enviem a l'activitat la posició de la llista que s'ha tocat mitjançant la variable "infoint" e iniciem l'activitat InfoMedicament assignada al intent "int2" amb startActivityForResult(). El codi "code1" és l'altre que hem declarat abans. Ens serveix per diferenciar aquesta activitat que llancem ara amb la de AfegirMedicament quan tornem amb el mètode onActivityResult(). A la imatge anterior podem veure tot el que acabo de descriure.

A partir d'aquí, se'ns obrirà la nova pantalla:




```
txtnom = (TextView) findViewById(R.id.txtnom);
txtcapsules = (TextView) findViewById(R.id.txtcapsules);
txtdosi = (TextView) findViewById(R.id.txtdosi);
txtfreq = (TextView) findViewById(R.id.txtfreq);
txtdurada = (TextView) findViewById(R.id.txtdurada);
txthora = (TextView) findViewById(R.id.txthora);
txtenrere = (TextView) findViewById(R.id.txtenrere);

position = getIntent().getExtras().getInt("pos");

SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
String txt1 = prefs.getString("nom" + position, "default");
String txt2 = prefs.getString("capsules" + position, "default");
String txt3 = prefs.getString("dosi"+position, "default");
String txt4 = prefs.getString("frequencia"+position, "default");
String txt5 = prefs.getString("durada"+position, "default");
String dia = prefs.getString("dia"+position, "0");
String hores = prefs.getString("hores"+position, "00");
String minuts = prefs.getString("minuts"+position, "00");
```

La informació es mostra en textViews dinàmics els quals assignem mitjançant la ID (txtnom, txtcapsules...). A la variable "position" recuperem l'Extra enviat amb l'intent d'abans que conté la posició de la llista que ha tocat l'usuari. Ara només haurem de guardar en variables les SharedPreferenced adequades en funció de la posició que li hem enviat:

```
String txt1 = prefs.getString("nom" + position, "default");
```

Si la posició val 2 per exemple, buscarà les preferències amb els noms "nom2, capsules 2..." i les assignarà a variables String per separat.

Un cop tenim aquesta informació la mostrem als textViews dinàmics amb el mètode "nomTextView.setText(String)":

```
txtnom.setText(txt1);
txtcapsules.setText(txt2);
txtdosi.setText(txt3);
txtfreq.setText(txt4);
txtdurada.setText(txt5);
```

D'aquesta manera aconseguim mostrar la informació del medicament situat a la posició que s'ha tocat a la pantalla principal. Si tornem amb el botó d'aquesta activitat "Tornar" i toquem un altre medicament, s'obrirà aquesta pantalla amb la informació del medicament que acabem de tocar perquè tots aquests mètodes s'executen de nou.

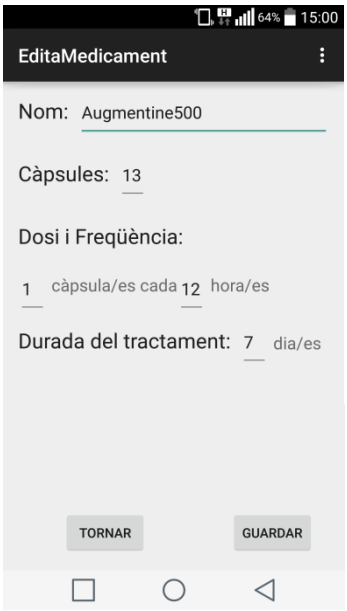
Des d'aquesta activitat tenim tres accions per fer:

1-Tornar: Funciona igual que el botó de AfegirMedicament. Finalitza l'activitat amb RESULT_CANCELED i finish() sense canvis.

2-Editar: Aquest botó inicia una nova activitat anomenada "EditaMedicament" que és una barreja entre InfoMedicament i AfegirMedicament. Se li envia la posició de la llista per saber de quin medicament es tracta i mostra la informació dins d'editors de text. La diferència amb InfoMedicament és que aquests editors es poden modificar i sobreescriure, per tant, també necessitarem els mètodes de SharedPreferences utilitzats a AfegirMedicament.

Veiem fragments de codi i resultats similars als vistos fins ara amb diferents activitats:

```
btedita.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent int1 = new Intent(InfoMedicament.this, EditaMedicament.class);
        int1.putExtra("pos", position);
        startActivityForResult(int1, code2);
    }
});
```




```
position = getIntent().getExtras().getInt("pos");

SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
String txt1 = prefs.getString("nom"+position, "default");
String txt2 = prefs.getString("capsules"+position, "default");
String txt3 = prefs.getString("dosi"+position, "default");
String txt4 = prefs.getString("frequencia"+position, "default");
String txt5 = prefs.getString("durada"+position, "default");

et1.setText(txt1);
et2.setText(txt2);
et3.setText(txt3);
et4.setText(txt4);
et5.setText(txt5);
```

Quan guardem s'executa el mètode semblant al utilitzat durant AfegirMedicament:

```
bt1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (et1.getText().toString().equals("")||et2.getText().toString().equals("")
                ||et3.getText().toString().equals("")||et4.getText().toString().equals("")
                ||et5.getText().toString().equals("")) {
            Toast.makeText(getApplicationContext(), "Emplena tots els camps", Toast.LENGTH_SHORT).show();
        } else {
            Intent intent = new Intent();
            String txt1=et1.getText().toString();
            String txt2=et2.getText().toString();
            String txt3=et3.getText().toString();
            String txt4=et4.getText().toString();
            String txt5=et5.getText().toString();
            SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("nom"+position, txt1);
            editor.putString("capsules"+position, txt2);
            editor.putString("dosi"+position, txt3);
            editor.putString("frequencia"+position, txt4);
            editor.putString("durada"+position, txt5);
            editor.commit();
            setResult(RESULT_OK, intent);
            finish();
        }
    }
});
```

L'única diferència és que agafa el text d'objectes EditText en comptes de TextView. Al guardar els canvis, veiem que s'executa el finish() de l'activitat amb RESULT_OK. Aquesta informació la rep el mètode onActivityResult() de InfoMedicament i també es finalitza per tornar a la pantalla principal. Aquest procés es realitza de tal forma que al guardar des de EditaMedicament, es torna a la pantalla principal amb la informació canviada.

Veiem el mètode onActivityResult de InfoMedicament:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    //comprovem que el resultat ha sigut "ok"
    if(requestCode==1236 && resultCode==RESULT_OK){
        Intent intent = new Intent();
        setResult(RESULT_OK, intent);
        finish();
        //Missatge informatiu per fer veure a l'usuari que s'ha editat correctament
        Toast.makeText(getApplicationContext(), "Editat correctament", Toast.LENGTH_SHORT).show();
    }
    /*si obté resultat "canceled", indiquem amb un toast que no fem canvis. Aixo passa al clicar el botó cancel·lar de la
     segona activitat o amb el botó retornar del mòbil*/
    if(requestCode==1236 && resultCode==RESULT_CANCELED){
        Toast.makeText(getApplicationContext(), "S'ha tornat sense canvis", Toast.LENGTH_SHORT).show();
    }
```

Si el resultat és RESULT_CANCELED es mostra un missatge conforme s'ha tornat sense canvis i el la informació dels textView sobre el medicament no haurà canviat, tornarà al estat que estava abans del startActivityForResult(). El cas contrari en que el resultat és RESULT_OK, generem un nou intent per assignar RESULT_OK a l'activitat principal i acabem aquesta activitat amb finish(). Aquest últim RESULT_OK és perquè l'activitat InfoMedicament havia estat cridada primerament amb un startActivityForResult() des de l'activitat principal i, per tant, sempre ha de rebre una resposta.

Ara entendrem el funcionament del codi que enviàvem amb cada intent en els startActivityForResult(). L'activitat principal pot cridar dos activitats: AfegirMedicament e InfoMedicament. Quan es retorna de qualsevol de les dos activitats, s'executa el mètode onActivityResult(). Ens interessa fer coses diferents en funció de quina activitat ha retornat informació. Per tant, posem condicions que faran que s'executi un codi o un altre en funció del "code" o "code1" enviat al començament:

```
if(requestCode==1235 && resultCode==RESULT_OK){
    //recupero la informació de la activitat on l'usuari ha editat el medicament
    SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
    String txt1 = prefs.getString("nom"+infoint, "default");
    String txt2 = prefs.getString("capsules"+infoint, "default");
    String txt3 = prefs.getString("dosi"+infoint, "default");
    String txt4 = prefs.getString("frequencia"+infoint, "default");
    String txt5 = prefs.getString("durada"+infoint, "default");
    strArr.set(infoint, txt1);
    adapter.notifyDataSetChanged();
}
/*si obté resultat "canceled", indiquem amb un toast que no fem canvis. Aixo passa al clicar el botó cancel·lar de la
 segona activitat o amb el botó retornar del mòbil*/
if(requestCode==1235 && resultCode==RESULT_CANCELED){
    Toast.makeText(getApplicationContext(), "S'ha tornat sense canvis", Toast.LENGTH_SHORT).show();
}
```

Aquestes condicions formen part de onActivityResult() de l'activitat principal i comproven si el codi que l'ha sol·licitat és el de "code"(1234) o "code1(1235).

1234 era del codi de AfegirMedicament i 1235 el de InfoMedicament. Així podem controlar les nostres accions en funció de l'activitat que ens acaba de donar una resposta.

3-Prendre: Aquesta és l'opció de tota l'aplicació amb més funcionalitats. Comencem per explicar la seva funcionalitat:

Al tocar el botó, es guarda la data i es mostra en un textView dinàmic prèviament buit. La data serveix al usuari per saber l'última vegada que s'ha pres aquell medicament. Acte seguit, es desactiva el botó perquè l'usuari no pugui tornar a prendre el medicament i s'activa un temporitzador en un altre textView dinàmic prèviament buit. Fins que el temporitzador no arriba a zero, l'usuari no pot tornar a prendre el medicament perquè el botó es queda desactivat. Per saber el temps que no pot prendre el medicament, el temporitzador agafarà la variable de la freqüència que s'ha de prendre l'usuari aquell medicament. Si són vuit hores, el temporitzador arribarà a zero passades les vuit hores.

Es modifica el valor de les càpsules perquè en resti una cada cop que es pren la medicació. Aquest valor s'actualitza a les SharedPreferences del medicament en qüestió.

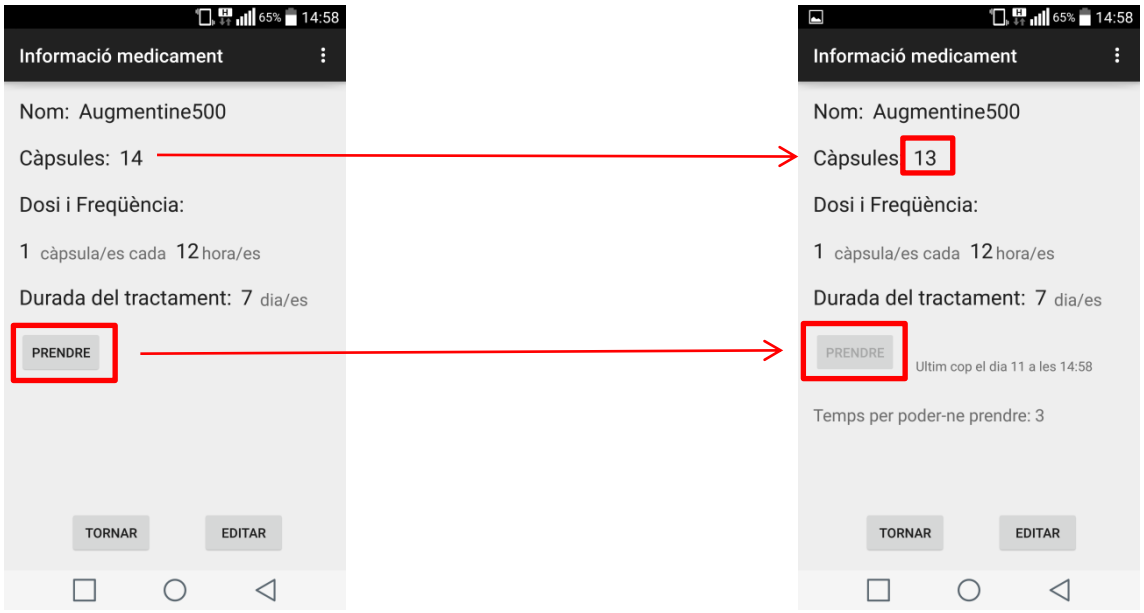
Hi ha un sistema de notificacions que avisa l'usuari en dos casos concrets:

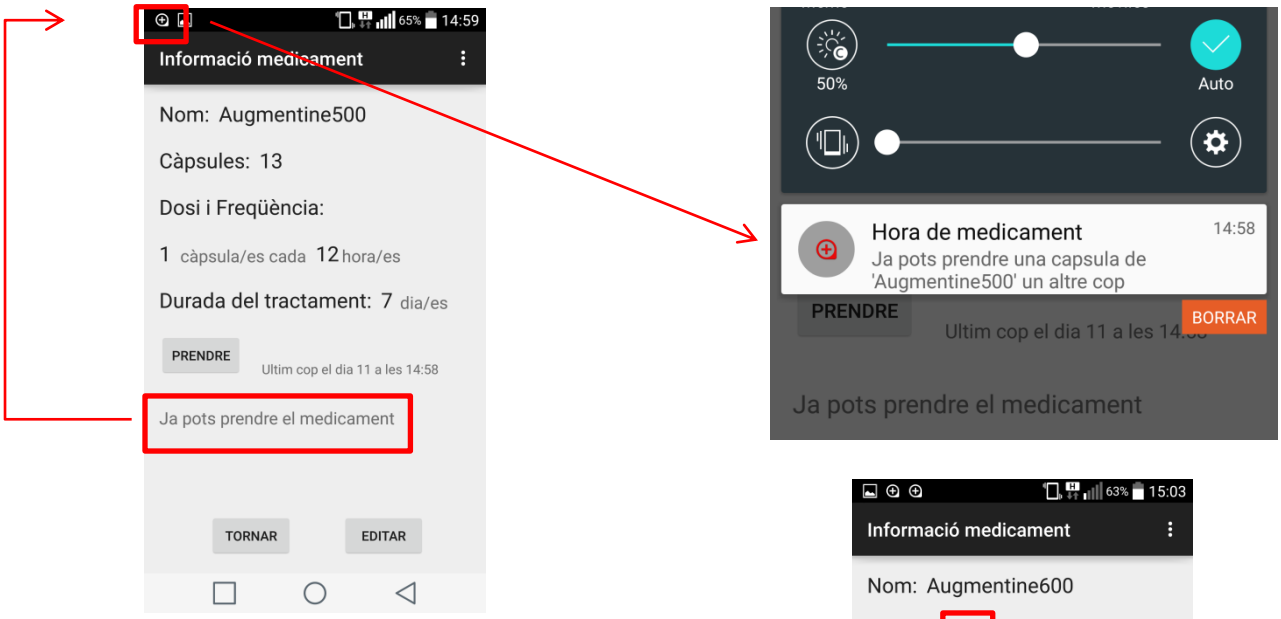
-Quan li queden tres càpsules o menys per informar-li que ha d'anar a comprar o simplement que s'està quedant sense aquella medicació.

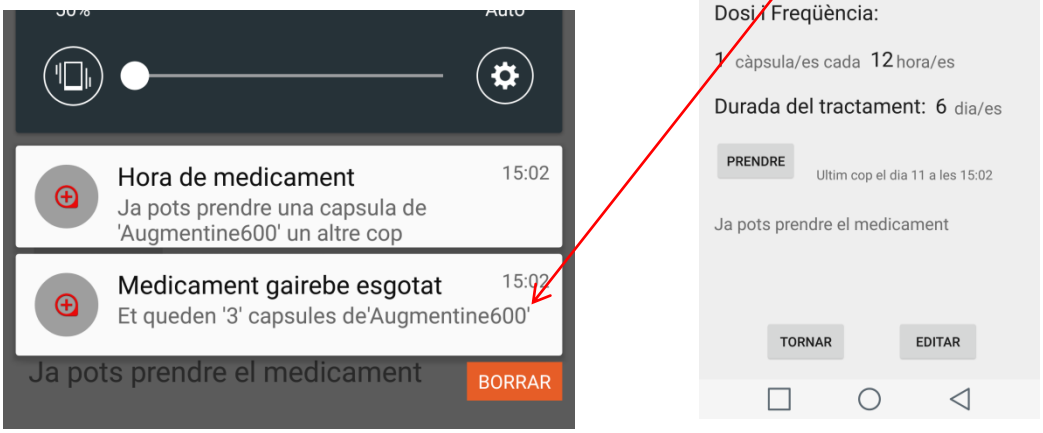
-Quan el temporitzador arriba a zero per avisar-lo i així saber quan pot tornar a prendre aquell medicament.

Si el valor de les càpsules és zero, tocar el botó mostrarà un missatge informatiu conforme t'has quedat sense aquell medicament i no es farà ninguna de les accions mostrades anteriorment.

Observem amb captures la funcionalitat:


Informació medicament
Nom: Augmentine500
Càpsules: 14
Dosi i Freqüència:
1 càpsula/es cada 12 hora/es
Durada del tractament: 7 dia/es
PRENDRE
TORNAR
EDITAR
Informació medicament
Nom: Augmentine500
Càpsules 13
Dosi i Freqüència:
1 càpsula/es cada 12 hora/es
Durada del tractament: 7 dia/es
PRENDRE
Ultim cop el dia 11 a les 14:58
Temps per poder-ne prendre: 3
TORNAR
EDITAR


Informació medicament
Nom: Augmentine500
Càpsules: 13
Dosi i Freqüència:
1 càpsula/es cada 12 hora/es
Durada del tractament: 7 dia/es
PRENDRE
Ultim cop el dia 11 a les 14:58
Ja pots prendre el medicament
TORNAR
EDITAR
50%
Auto
Hora de medicament
14:58
Ja pots prendre una capsula de 'Augmentine500' un altre cop
PRENDRE
BORRAR
Ja pots prendre el medicament


Hora de medicament
15:02
Ja pots prendre una capsula de 'Augmentine600' un altre cop
Medicament gairebe esgotat
Et queden '3' capsules de'Augmentine600'
Ja pots prendre el medicament
BORRAR
Informació medicament
Nom: Augmentine600
Càpsules: 3
Dosi i Freqüència:
1 càpsula/es cada 12 hora/es
Durada del tractament: 6 dia/es
PRENDRE
Ultim cop el dia 11 a les 15:02
Ja pots prendre el medicament
TORNAR
EDITAR

Cal remarcar que independentment de la pantalla en que es trobi l'usuari en el moment que el temporitzador arriba a zero, rebràs la notificació avisant-te que ja pot prendre un altre cop el medicament qui sigui.

Veiem el mètode onClick() del botó "Prendre" per veure el codi de tot aquest funcionament:

```
btnprendre.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Calendar c = Calendar.getInstance();
        int minuts = c.get(Calendar.MINUTE);
        int hores = c.get(Calendar.HOUR_OF_DAY);
        int dia = c.get(Calendar.DAY_OF_MONTH);
        txthora.setText(String.format("Ultim cop el dia %02d a les %02d:%02d",dia,hores,minuts));
        btnprendre.setEnabled(false);

        new CountDownTimer(5000, 1000) {

            public void onTick(long millisUntilFinished) {
                txtenrere.setText("Temps per poder-ne prendre: " + millisUntilFinished / 1000);
            }

            public void onFinish() {
                notificacioEnrere();
                txtenrere.setText("Ja pots prendre el medicament");
                btnprendre.setEnabled(true);
            }
        }.start();

        SharedPreferences prefs = getSharedPreferences("MisPreferencias", Context.MODE_PRIVATE);
        String scapsules = prefs.getString("capsules" + position, "default");
        int capsules = Integer.parseInt(scapsules);
        capsules = capsules-1;
        if(capsules<=3&&capsules>0){
            notificacioCasiAcabat();
        }
        if(capsules<0){
            Toast.makeText(getApplicationContext(), "Medicament esgotat", Toast.LENGTH_SHORT).show();
        }else {
            scapsules = Integer.toString(capsules);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("capsules" + position, scapsules);
            editor.putString("dia"+position,Integer.toString(dia));
            editor.putString("hores"+position,Integer.toString(hores));
            editor.putString("minuts"+position,Integer.toString(minuts));
            editor.commit();
            txtcapsules.setText(scapsules);
        }
```

Quan es toca el botó, es crea un objecte "Calendar" que agafa l'hora del sistema mitjançant "Calendar.getInstance()". Assignem els valors minuts, hora del dia i dia del més a tres variables enteres. Podem fer-ho gràcies al objecte Calendar que hem creat:

```
int minuts = c.get(Calendar.MINUTE);
```

Utilitzant aquestes variables ja podem indicar al textView de la data de l'última presa amb setText() i desactivem el botó Prendre amb "setEnabled(false)".

Després es crea el temporitzador (les variables estan en milisegons). Està posat perquè duri cinc segons per no haver d'esperar les hores que indiqui la variable de la freqüència per veure el funcionament. A la versió definitiva, hauríem de posar el valor de "txtfrequencia.getText().toString()" i passar-lo a milisegons. El temporitzador té dos mètodes propis, onTick() i onFinish(). Per cada "tick" del temporitzador s'actualitza el text del textView que conté el temporitzador per mostrar quants segons queden en cada moment (també es podria calcular en minuts i/o hores). Al arribar a zero i acabar, s'executa un mètode "notificacióEnrere()" que he creat jo i actualitza el text del textView per indicar que ja es pot tornar a prendre el medicament. Finalment, activem de nou el botó amb setEnabled(true).

Fem una ullada als mètodes de notificacions que he creat perquè saltin notificacions sempre que ens interessi:

```
private void notificacioCasiAcabat(){
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
    builder.setContentTitle("Medicament gairebe esgotat");
    int intcapsules = Integer.parseInt(txtcapsules.getText().toString())-1;
    builder.setContentText("Et queden '"+intcapsules+"' capsules de '"+txtnom.getText()+"'");
    builder.setSmallIcon(R.mipmap.ic_launcher);
    builder.setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION));
    builder.setDefaults(Notification.DEFAULT_VIBRATE);

    Notification notificacio = builder.build();
    NotificationManager manager = (NotificationManager)this.getSystemService(NOTIFICATION_SERVICE);
    manager.notify(8,notificacio);
}

private void notificacioEnrere(){
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
    builder.setContentTitle("Hora de medicament");
    builder.setContentText("Ja pots prendre una capsula de '"+txtnom.getText()+"' un altre cop");
    builder.setSmallIcon(R.mipmap.ic_launcher);
    builder.setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION));
    builder.setDefaults(Notification.DEFAULT_VIBRATE);

    Notification notificacio = builder.build();
    NotificationManager manager = (NotificationManager)this.getSystemService(NOTIFICATION_SERVICE);
    manager.notify(9,notificacio);
}
```

"NotificationCompat.Builder" és una classe de la llibreria d'Android que permet crear notificacions personalitzades amb moltes de les possibilitats que permet el mètode "Builder.set...". En el meu cas, jo he creat dos de diferents, una per avisar que ja es pot tornar a prendre el medicament i l'altre per advertir que queden poques càpsules. El que canvia en tots dos casos és el text del títol i el subtítol. Com podem veure, el text del nom del medicament i les càpsules és dinàmic perquè agafa les variables corresponents a cada cas.

Tornem al temporitzador: ara només haurem d'executar-lo mitjançant "start()". El següent pas és accedir a les SharedPreferences per restar una càpsula a la variable de capsules restants. Un cop l'hem restat fem un parell de comprovacions:

-Si el número de càpsules és més petit o igual a tres però més gran que zero, s'enviarà la notificació "notificacioCasiAcabat()".

-Si el resultat de restar una càpsula és més petit que zero, s'informarà amb un Toast que el medicament està esgotat. En qualsevol altre cas afegirem el valor actual de càpsules al SharedPreferences. També hi guardarem tres variables noves per saber la data de l'última vegada que hem pres aquell medicament. Per aquest motiu, si tornem a l'activitat principal i seleccionem un altre medicament podrem veure dates diferents perquè han de ser guardades a algun lloc. D'aquesta manera tan simple i efectiva guardem més dades dinàmiques sense gaire dificultat ni consum de recursos. En acabar l'edició, executem el editor.commit() i mostrem el nou valor de càpsules al seu textView corresponent amb setText() tal i com es veu a la figura.

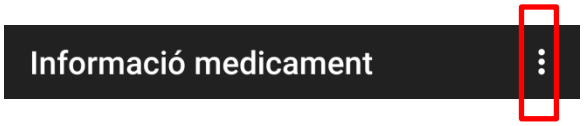
```
SharedPreferences.Editor editor = prefs.edit();
editor.putString("capsules" + position, scapsules);
editor.putString("dia"+position,Integer.toString(dia));
editor.putString("hores"+position,Integer.toString(hores));
editor.putString("minuts"+position,Integer.toString(minuts));
editor.commit();
txtcapsules.setText(scapsules);
```

5.5.2 Menú

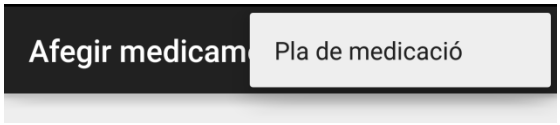
El funcionament dels menús de cada activitat el vaig ensenyar a la part de maquetació. Ara veurem el codi que es genera i com podem fer-lo servir per crear el nostre propi ítem i programar-lo. Tenim dos mètodes pel menú a cada activitat:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_info_medicament, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        Intent intmenu = new Intent(this,ActivityPrincipal.class);
        startActivity(intmenu);
        return true;
    }

    return super.onOptionsItemSelected(item);
```

Es crea el menú de forma similar a la llista. En comptes d'un adaptador es fa servir un "inflater" per generar espais per cada ítem que hi hagi al layout del menú. D'aquesta manera podem tenir un menú dinàmic des de la tecla de la barra d'eines:



Al tocar sobre el menú, ens apareixen el items que tinguem. En el meu cas no tenia cap funcionalitat a aplicar-li al menú i vaig decidir posar un accés directe a l'activitat principal per les altres tres activitats. D'aquesta manera l'usuari pot tornar al seu pla de medicació també per aquesta via.



Tal i com es veu a la figura del codi, només haurem de crear un intent quan es detecta que la ID del item que s'ha tocat és la correcte (només tenim una de totes maneres) i executar-lo amb startActivity() per conduir a l'activitat principal.

```
Intent intmenu = new Intent(this,ActivityPrincipal.class);
startActivity(intmenu);
```

5.5.3 Importacions

Tots els objectes que fem servir ja siguin de la llibreria d'Android o de Java s'han de importar per poder funcionar. En cas contrari, el programa no es deixa executar. Els "imports són totalment necessaris i Android Studio permet minimitzar-los perquè no molestin tant estèticament.

```
import android.app.Notification;
import android.app.NotificationManager;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.media.RingtoneManager;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.support.v4.app.NotificationCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import java.util.Calendar;
```

Aquest exemple són tots els imports de la classe InfoMedicament. Per totes les classes i activitats haurem d'importar els elements que fem servir, és a dir, no serveix importar-lo només a una classe.

6. Possibles funcionalitats noves

Pillbox va néixer per facilitar la feina a les persones que s'encarreguen de gestionar algun tipus de pastiller per gent gran o als individus amb plans de medicació extensos, com ara malalts crònics. Qualsevol funció que compleixi aquest objectiu és benvingut per afegir a l'aplicació. Idees en tenia moltes des de un principi, però les limitacions tant per coneixements com per falta d'elements, moltes funcionalitats no han pogut veure la llum. M'agradaria que tingues més opcions com un pla de medicació escrit. A continuació, comentaré possibles millores i plantejaré possibles projectes ja siguin per mi o per altres alumnes.

6.1 Funcionalitats extres per l'aplicació

-Poder afegir els camps "vigència", "comentaris", "prescriptor" i "centre". Aquest tipus d'informació seria més del metge que ha donat el pla de medicació al pacient. Es podrien afegir com a camps opcionals u obligatoris, en funció del control que vulguem tenir sobre les dades que introdueix l'usuari.

-Donar la possibilitat d'afegir un telèfon d'una tercera persona (normalment l'encarregat de gestionar el pla de medicació en cas de ser una persona gran el pacient) per avisar en cas que un medicament s'hagués saltat o no s'hagués pres.

-Poder afegir medicaments que no funcionen per càpsules, com per exemple cremes o sobres indicant prèviament el tipus de medicament que es tracta.

-Crear una base de dades més complexa amb moltes més dades per aprofitar l'ús excessiu en recursos que suposaria. Amb una base de dades SQLite seria més fàcil afegir més opcions pels medicaments i es podrien donar de baixar correctament. Utilitzant SharedPreferences, si donem de baixa un medicament i la llista es fa més petita, desordenaria els medicaments i la informació que mostraria la preferència "nom2" no seria realment el nom del "medicament 2". És una de les úniques avantatges que hi trobo per treballar amb base de dades i no amb preferències compartides.

-Intentar reduir la variable "durada del tractament" (en dies) cada cop que acabi un dia. Quan la variable d'un medicament arribi a zero, haurà acabat el tractament i automàticament es podria esborrar el medicament de la llista a la pantalla principal.

6.2 Projectes a considerar

-Si la relació metge-pacient-farmacèutic es pogués millorar gràcies a tràmits digitals, permetríem només amb el mòbil facilitar tots els processos. Per exemple, es podria investigar la manera de llegir el codi de barres amb el mòbil d'un pla de medicació imprès i afegir tots els medicaments a una aplicació al instant, els mateixos que acabem d'escanejar al paper. Una forma de fer-ho podria ser amb codis QR. Després arribaríem a la farmàcia, ensenyaríem el codi de barres i automàticament el farmacèutic podria veure tot el que necessites i el metge que t'ho ha receptat.

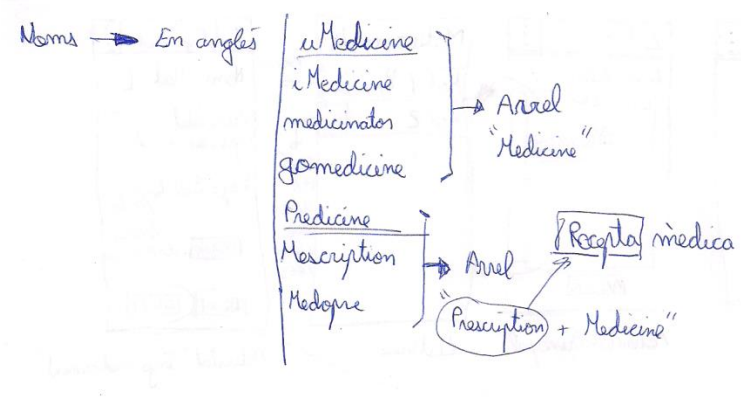
-Llegir una base de dades plena d'informació sobre medicaments per poder escollir-los en lloc de crear-los. Possiblement aquestes bases de dades s'haurien d'extreure d'alguna farmàcia i/o farmacèutica. D'aquesta manera, també podríem veure fotografies dels medicaments e informació i prospectes del mateix. També hi hauria més seguretat i prevenció a l'hora de prendre medicaments sota recepta, perquè el gestor de medicaments en si no controla si estàs barrejant medicaments o si n'estàs prenent masses.

-Investigar sobre els aparells per mòbil que permeten donar informació a través d'aplicacions sobre les teves constants vitals, temperatura corporal... fins i tot ecografies. Recollir i administrar tota aquesta informació a una aplicació com a projecte.

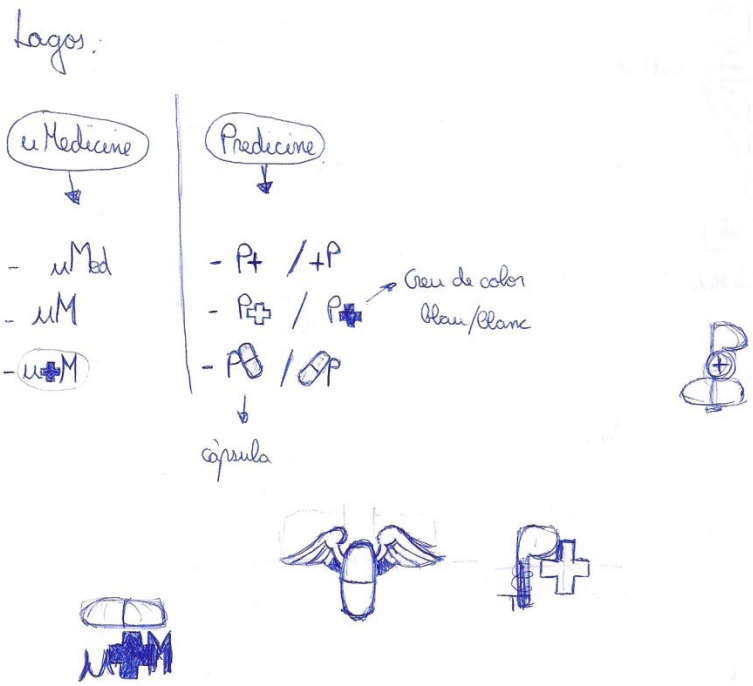
7. Evolució y Aspecte final

La meva aplicació va començar amb un nom i un logotip orientatiu. Vaig barrejar molts noms fins que em va agradar "Predicine". Predicine és un acrònim de "Prescription Medicine". Més tard, vaig estar pensant que no era exactament un nom gaire encertat perquè l'aplicació tenia més a veure amb la gestió dels medicaments i no de receptes mèdiques. Arribats a aquesta conclusió vaig centrar-me en la funció que fa l'aplicació en sí i se'm va acudir el nom de "Pillbox", que és una traducció literal de l'objecte físic que es fa servir per gestionar medicaments.

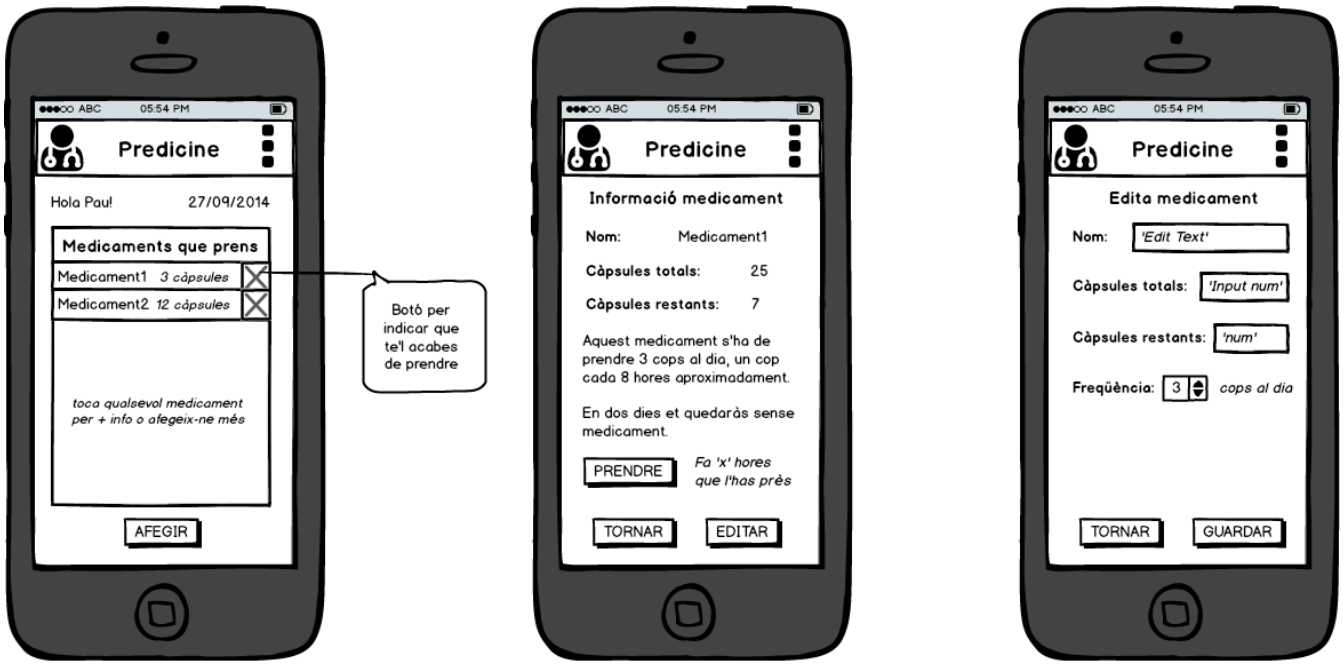
Veiem les captures de l'evolució d'esbossos i mockups bàsics per una correcte programació posterior.



Esbossos de possibles noms i logotips

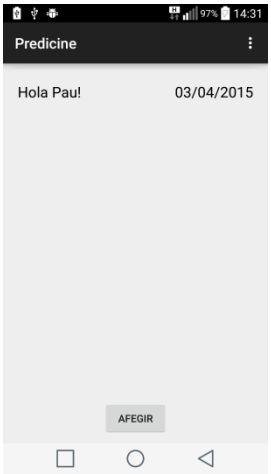


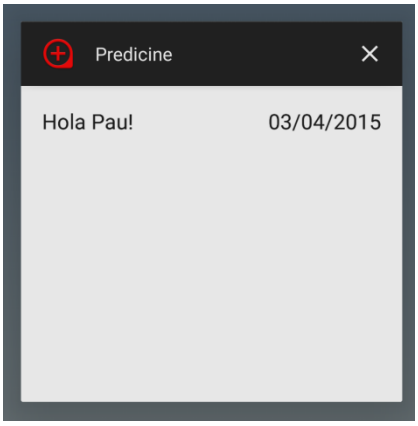
Logotip anterior amb fons transparent generat automàticament per Android Asset Studio, l'eina online descrita a la part de producció.



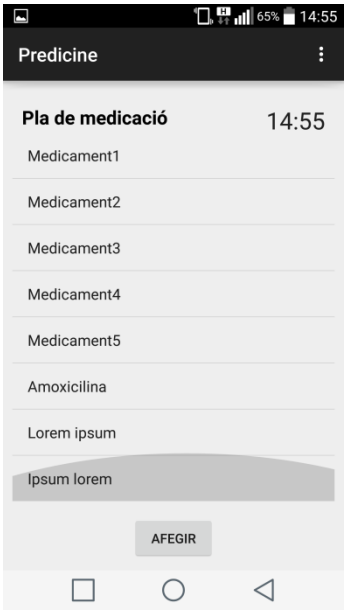
Activitat principal | Activitat "Info medicament" | Activitat "Edita medicament"

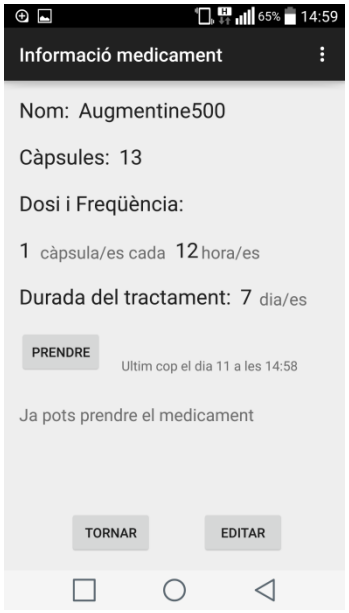
Mockups per fer servir de guia a l'hora de maquetar l'aplicació.





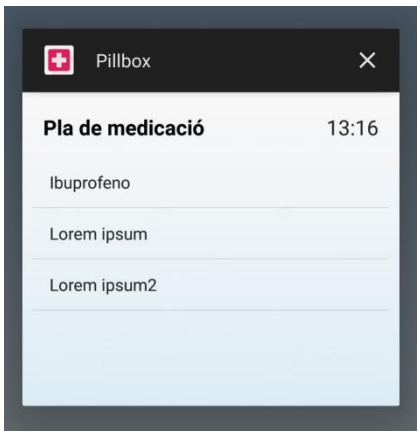
Primeres captures vistes a la part de maquetació.

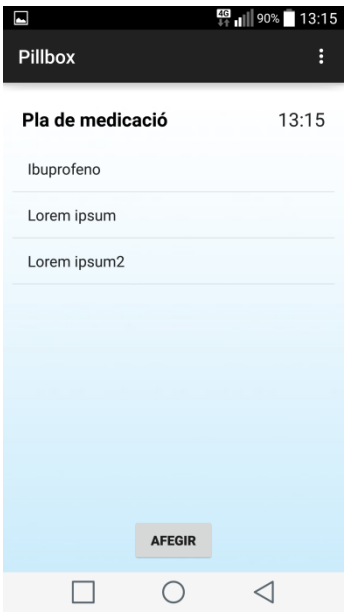


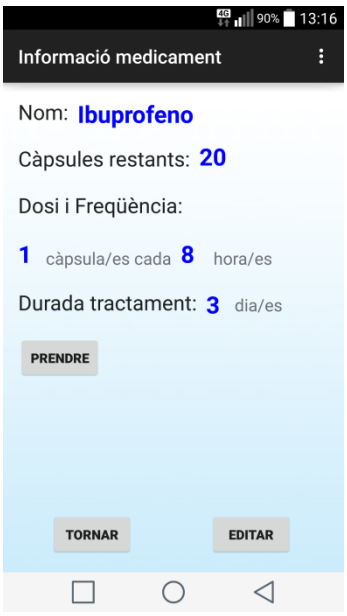


Captures amb l'aplicació programada amb el primer aspecte.

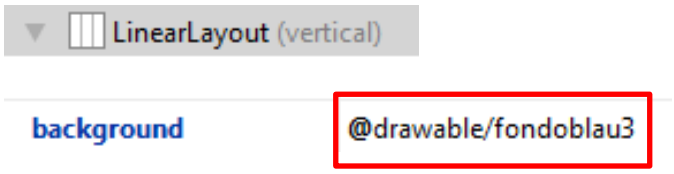
Finalment veurem algunes captures del aspecte actual:



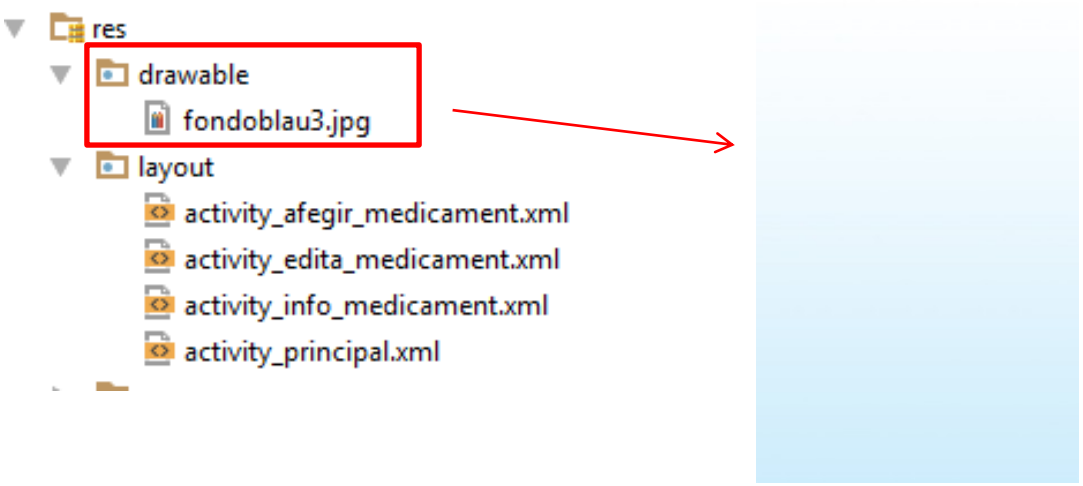




Principalment, ha canviat el logotip i el fons de l'aplicació. Per fer-ho, hem d'assignar les imatges al XML dels layouts:



Hem d'assignar als layouts contenidors la imatge de fons "fondoblau3" que ha d'estar a la carpeta "Drawable" dels recursos de la nostre aplicació:



El logotip, com ja sabem, hem la seva localització al Android Manifest:

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Pillbox"
    android:theme="@style/AppTheme" >
```

Posarem el nom també a l'etiqueta "label" mitjançant una variable del fitxer "Strings.xml":

```
<resources>
    <string name="app_name">Pillbox</string>
    <string name="labelPrincipal">Pillbox</string>
    <string name="action_settings">Pla de medicació</string>
    <string name="text1">Pla de medicació</string>
```

8. Conclusions

Després de gairebé vuit mesos interromputs per períodes alterns de feina i exàmens, estic content amb el treball. No em refereixo només a l'aplicació en sí. Aquest treball ha significat molt per mi, em va animar a introduir-me al món de la programació orientada a objectes. Encara que em quedi molt per aprendre, va ser el motiu pel qual vaig apuntar-me al Cicle Formatiu de Grau Superior d'Informàtica en el desenvolupament d'aplicacions multi plataforma. En aquest cicle es toca molt codi i em van ajudar a completar les bases essencials d'Android i Java que vaig començar a les optatives del quart curs a la universitat del Grau en Multimèdia. La programació és la meva passió i quan aprengui del tot la programació orientada a objectes segurament estudiaré un màster o postgrau orientat cap això mateix.

Cal destacar que jo començava pràcticament de zero en el desenvolupament Android. Tots els coneixements els he anat aprenent aquest últim any i l'aprenentatge autònom ha sigut enorme i molt útil. Aquest treball m'ha demostrat que encara que no sàpigues fer alguna cosa et pots defensar i a mesura que guanyes experiència mitjançant assaig-error, tot va sortint, a vegades de forma inesperada i a vegades amb resultats millors dels previstos. Tot i així em queda moltíssim per millorar i poder avançar, començant per les funcionalitats que m'hauria agradat posar a l'aplicació i no he pogut com les que he comentat al punt anterior. En definitiva, la millor manera d'aprendre a programar es així, programant. I res millor que un projecte com aquest per agafar destresa.

També vull remarcar una falta d'organització inicial que em va obligar a sol·licitar una pròrroga. Les classes del superior i la feina personal van fer impossible complir les dades establertes amb el treball de la universitat. De tota manera, no he tingut problemes amb l'organització de la feina del treball en sí. Només em va portar més temps del esperat la programació de certes funcionalitats de l'aplicació, com ara la de guardar dades sense base de dades.

En conclusió, ha sigut un projecte que m'ha fet aprendre e indagar per mi mateix molts coneixements. Els objectius inicials han estat complerts tot i que m'agradaria millorar algunes funcionalitats actuals per fer-les més usables i afegir-ne alguna més. És millorable en l'àmbit de disseny encara que jo m'he centrat en la programació. És més, jo seguiria treballant en el projecte per practicar més el codi i els mètodes Android més que per un disseny d'estils. Ha sigut un projecte per dir adéu a la carrera i espero tornar-hi aviat amb un màster o postgrau gràcies als ànims que he guanyat per arribar a ser programador.

9. Bibliografia

La bibliografia la he anat actualitzant al llarg de tot el projecte, cada cop que necessitava consultar alguns dubtes o simplement informació. Moltes de les consultes eren puntuals mentre que d'altres van ser reiterades en funció de necessitat. Aquí mostraré totes les referències que he consultat amb més o menys freqüència:

10/10/14 – Consultes per l'estudi de mercat: l'App de la Seguretat Social

http://www.elmundo.es/blogs/elmundo/applicate/2013/04/24/la-app-de-la-seguridad-social.html

http://www.seg-social.es/infosegsocialappmovil/index_page_id_10.html

http://www.seg-social.es/infosegsocialappmovil/index_page_id_12.html

http://www.seg-social.es/infosegsocialappmovil/index_page_id_13.html

10/02/15 – Consultes reiterades al diari "El Mundo": Secció de la tecnologia mèdica

http://www.elmundo.es/elmundosalud/tecnologiamedica.html

10/02/15 – Article del diari "El Mundo" sobre una empresa espanyola que ajuda a millorar la qualitat de vida als pacients crònics gràcies a la tecnologia, permetent un bon seguiment

http://www.elmundo.es/andalucia/2015/03/16/5505c99aca4741a2048b456c.html

11/02/15 – Article web en anglès sobre la tecnologia y la salut

http://health.usnews.com/health-news/hospital-of-tomorrow/articles/2013/07/12/how-technology-is-transforming-health-care

12/02/15 – Article web en anglès sobre les innovacions tecnològiques sanitàries l'any 2014

https://getreferralmd.com/2013/11/health-care-technology-innovations-2013-infographic/

12/02/15 – Article web en anglès sobre les innovacions tecnològiques sanitàries l'any 2015

https://getreferralmd.com/2015/02/the-10-biggest-innovations-in-health-care-technology-in-2015/

29/06/15 - Tutorial ListView Android (Part 1 i Part2):

https://www.youtube.com/watch?v=BSZLqBWKTHw

02/07/15 - Tutorial Dynamic ListView:

https://www.youtube.com/watch?v=tNoeFkXCZ6w

03/07/15 – Almacenamiento de datos en Android @UPV (Universitat Politècnica de València):

https://www.youtube.com/watch?v=Zgb3Z_Czunk&index=52&list=PL6kQim6ljTJvgZ8RiRSA5zUfpOjY6NYDv

http://www.androidcurso.com/index.php/tutoriales-android/42-unidad-9-almacenamiento-de-datos/297-alternativas-para-guardar-datos-permanentemente-en-android

05/07/15 – Notificaciones en Android:

http://www.sgoliver.net/blog/notificaciones-en-android-ii-barra-de-estado/

http://www.vogella.com/tutorials/AndroidNotifications/article.html

http://www.tutorialspoint.com/android/android_notifications.htm

https://www.youtube.com/watch?v=gbVYApHH9Hk

(Versió API 11 o menor de les Notificacions):
https://www.youtube.com/watch?v=HcE1_SKmWW8&index=51

06/07/15 – Uso de Base de Datos en Android @UPV

https://www.youtube.com/watch?v=J2W862omYqk&index=55

06/07/15 – Tutorial de Base de Datos SQLite en aplicaciones Android

http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/

10/07/15 – Preferencias en Android I: Shared Preferences

http://www.sgoliver.net/blog/preferencias-en-android-i-shared-preferences/

10/07/15 – Preferencias en Android II: PreferenceActivity

http://www.sgoliver.net/blog/preferencias-en-android-ii-preferenceactivity/

10/07/15 – StackOverFlow: Consultes sobre comunicació entre activitats

http://stackoverflow.com/questions/5944503/android-getintent-getextras-returns-null

http://stackoverflow.com/questions/28981935/attempt-to-invoke-virtual-method-on-a-null-object-reference

09/08/15 – Android Developers Documentation: CountDownTimer

http://developer.android.com/reference/android/os/CountDownTimer.html

09/08/15 – Android Application Development Tutorial – 87 – Getting the Time from the System

https://www.youtube.com/watch?v=N2Tx8S2V8ek

09/08/15 – StackOverFlow: Consultes sobre l'obtenció de la hora del sistema

http://stackoverflow.com/questions/5369682/get-current-time-and-date-on-android

12/08/15 – Consultes sobre formats de cadenes de text (String format):

http://developando.com/blog/java-formatear-cadenas-string-format

10. Annexos

10.1 Google Play (Play Store)

https://play.google.com/store/apps/details?id=com.tfg.pau.tfg

10.2 Calendari

	Nombre de tarea	Duración	Comienzo	Fin
1	**Treball de Final de Grau**	**244 días**	**lun 06/10/14**	**jue 10/09/15**
2	**Memòria**	**244 días**	**lun 06/10/14**	**jue 10/09/15**
3	**1. Introducció**	**244 días**	**lun 06/10/14**	**jue 10/09/15**
4	1.1 Objectius	1 día	lun 06/10/14	lun 06/10/14
5	1.2 Calendari	243 días	mar 07/10/14	jue 10/09/15
6	**2. Filosofia de la tecnologia**	**3 días**	**mar 10/02/15**	**jue 12/02/15**
7	2.1 Introducció	1 día	mar 10/02/15	mar 10/02/15
8	2.2 Tecnologia vinculada sanitat	2 días	mié 11/02/15	jue 12/02/15
9	**3. Estudi de mercat**	**11 días**	**mar 07/10/14**	**mar 21/10/14**
10	3.1 Introducció	1 día	mar 07/10/14	mar 07/10/14
11	3.2 Objectiu investigació	1 día	mié 08/10/14	mié 08/10/14
12	3.3 Anàlisi situació actual	2 días	vie 10/10/14	lun 13/10/14
13	3.4 Anàlisi DAFO	5 días	mar 14/10/14	lun 20/10/14
14	3.5 Públic objectiu	1 día	mar 21/10/14	mar 21/10/14
15	3.6 Referents	1 día	mar 21/10/14	mar 21/10/14
16	**4. Producció**	**118 días**	**lun 23/03/15**	**mié 02/09/15**
17	4.1 IDE Android Studio	3 días	lun 23/03/15	mié 25/03/15
18	4.2 Accions prèvies	2 días	jue 26/03/15	vie 27/03/15
19	4.3 Creació d'una activitat	3 días	lun 30/03/15	mié 01/04/15
20	4.4 Maquetació	7 días	lun 15/06/15	mar 23/06/15
21	**4.5 Programació**	**13 días**	**lun 17/08/15**	**mié 02/09/15**
22	4.5.1 Activitats	10 días	lun 17/08/15	vie 28/08/15

	Nombre de tarea	Duración	Comienzo	Fin
22	4.5.1 Activitats	10 días	lun 17/08/15	vie 28/08/15
23	4.5.2 Menú	2 días	lun 31/08/15	mar 01/09/15
24	4.5.3 Importacions	1 día	mié 02/09/15	mié 02/09/15
25	**5. Idees per funcionalitats noves**	**3 días**	**jue 03/09/15**	**lun 07/09/15**
26	5.1 Funcionalitats extres	2 días	jue 03/09/15	vie 04/09/15
27	5.2 Projectes a considerar	1 día	lun 07/09/15	lun 07/09/15
28	**6. Evolució i aspecte final**	**2 días**	**lun 07/09/15**	**mar 08/09/15**
29	**7. Conclusions**	**1 día**	**lun 06/10/14**	**lun 06/10/14**
30	**8. Bibliografia**	**239 días**	**vie 10/10/14**	**mié 09/09/15**
31	**9. Annexos**	**1 día**	**jue 10/09/15**	**jue 10/09/15**
32	**Aplicació**	**223 días**	**lun 06/10/14**	**mié 12/08/15**
33	**1. Idea**	**7 días**	**lun 06/10/14**	**mar 14/10/14**
34	**2. Esbossos i Mockups**	**3 días**	**mié 15/10/14**	**vie 17/10/14**
35	**3. Maquetació**	**10 días**	**lun 29/06/15**	**vie 10/07/15**
36	3.1 Layouts XML	7 días	lun 29/06/15	mar 07/07/15
37	3.2 Android Manifest	2 días	mié 08/07/15	jue 09/07/15
38	3.3 Menú	1 día	vie 10/07/15	vie 10/07/15
39	3.4 Recursos	7 días	lun 29/06/15	mar 07/07/15
40	**4. Programació**	**26 días**	**mié 08/07/15**	**mié 12/08/15**
41	4.1 Classes Java - Activitats	26 días	mié 08/07/15	mié 12/08/15
42	4.2 Intents	5 días	mié 08/07/15	mar 14/07/15
43	4.3 Funcionalitats de las pantalles	7 días	mié 15/07/15	jue 23/07/15
44	4.4 Shared Preferences	4 días	vie 10/07/15	mié 15/07/15
45	4.5 Integració de tots els recursos	2 días	vie 24/07/15	lun 27/07/15